ON-LINE GRAPHICAL DISPLAY SYSTEM

by

William Douglas Bechtel

UNITED STATES
NAVAL POSTGRADUATE SCHOOL

THESIS

ON-LINE GRAPHICAL

DISPLAY SYSTEM

by

William Douglas Bechtel

June, 1968

ON-LINE GRAPHICAL

DISPLAY SYSTEM

by

William Douglas Bechtel
Lieutenant, United States Coast Guard
B.S., Coast Guard Academy, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June, 1968

ABSTRACT

On-line graphical display of data of various types offers a powerful tool for the visual analysis of information. A software system is proposed to implement on-line display in a general purpose hybrid simulation laboratory. The subroutines required to implement this are put forward. Specific proposals are made in areas related to the basic software package. An application example is included to show the interrelationships of the software package.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FLOWCHARTS

ACKNOWLEDGEMENTS

The author wishes to take this opportunity of expressing his appreciation to the many people who assisted him in the preparation of this thesis. The writer is especially indebted to his advisor, Professor Mitchell L. Cotton, whose helpful guidance and counsel has proved invaluable. The author would also like to thank Mr. Robert L. Limes who never seemed to tire of helping the author with the programming problems. Thanks are also in order for the rough draft typist whose assistance the author could never have afforded if he were not married to her.

W. D. B.

CHAPTER I

INTRODUCTION

Graphical presentation of data is becoming a major application of digital computers. Quite often, graphic display is used in the solution of problems since the communication between man and display occurs in man's real time. Presented here is the nucleus of a software package which will display data and provide a basic man-machine communication link. This system will provide a useful analysis and design tool.

Background

The software system to be proposed will enable the display unit to act as an input/output device for the user's program in memory. Thus, the system must be completely compatible with the present software environment and does not allow any modification of the present software.

Considerable investigation in the field of digital display has been carried on at Massachusetts Institute of Technology and General Motors Corporation Research Laboratories.

Sketchpad[1] developed at M.I.T. is a sophisticated system which operates on data to permit highly flexible man-machine communications.

---

[1] Ivan E. Sutherland, "Sketchpad - A Man-machine Graphical Communication System," Proceedings of the Spring Joint Computer Conference (1963), p. 329.

For example, given three points on the display area, a circle can be made to pass through the three points. To perform these operations, a relatively complex data storage scheme is used involving multilevel relationships within tree-ordered structures. DAC-1[2,3] developed at the G.M. Research Laboratories provides for Designed Augmented by Computer. It is used for analysis and design of mechanical systems. The computer and display hardware and software were designed specifically for the design task. While DAC-1 is a sophisticated design tool, it lacks the versatility of the Sketchpad.

Both of the systems described above require a much larger memory allocation than is available in the Naval Postgraduate School's SDS 930 Computer.

Many other display systems have been developed;[4,5,6] however, most of these use separate small scale computers to provide display functions.

---

[2]Edwin L. Jacks, "A Laboratory for the Study of Graphical Man-machine Communication," Proceedings of the Fall Joint Computer Conference (1964), p. 343.

[3]Thomas R. Allen and James E. Foote, "Input/output Software Capability for a Man-machine Communication and Image Processing System," Proceedings of the Fall Joint Computer Conference (1964), p. 387.

[4]N. A. Ball et al., "A Shared Memory Computer Display," Institute of Electrical and Electronics Engineers Transactions on Electronics Computers (October, 1966), p. 750.

[5]R. H. Terlet, "The CRT Display Subroutines of the IBM 1500 Instructional System," Proceedings of the Fall Joint Computer

Goals and Example of Use

The goal of this system is to provide a simple to use, yet versatile, software package that can be used in a variety of situations.

An example of the use of this system in war games follows: the war game consists of a destroyer against a PT boat. The conning officer of each ship sits at a display console and controls the course and speed of his ship and directs his ship's weapons. The display shows both ships and pertinent information. Shell trajectories are computed by the digital computer and splash points are displayed. The analog computer simulates the characteristics of each ship and weapon.

Hardware Organization

The display console is made up of four principle elements as follows:

1. A large cathode ray tube,
2. An alphanumeric keyboard,
3. A function switch keyboard, and
4. A light pen.

Each of the two display units has its own interface with the digital computer so they can be operated individually and independently.

Conference(1967), p. 169.

[6]William H. Ninke, "Graphic 1 - A Remote Graphical Display System," Proceedings of the Fall Joint Computer Conference(1965), p. 839.

Basic Features

The concept of versatility provided the main criteria for the software package. For this reason, several low level subroutines were used rather than one large subroutine. The user has complete control over all parameters, with few arbitrary decisions by the author. This leads to quite long calling sequences in some cases, but adds to the versatility of the system. The subroutines are written in assembly language and callable by Fortran IV, although slight modifications would make them callable by any desired language. The basic unit of the display console is the raster unit. A raster unit is 1/2048 of the horizontal cathode ray tube width and 1/2048 of the vertical cathode ray tube height. All data generated by the software package is in addition to the user's data. Thus, the user's original data is never destroyed.

In addition to displaying vectors, points, and alphanumeric data, the software package can draw curves, change the curves, and delete any curves or portions of the curves. Plotting of data is not limited to cartesian coordinates, but polar coordinates may also be used.

CHAPTER II

DESCRIPTION OF SYSTEM ENVIRONMENT

Display Unit

The viewing area of the display unit is a nominal twenty-three inch diagonal CRT with electrostatic focusing and magnetic deflection. An area of eighteen inches horizontally by thirteen inches vertically is available for display. The display unit operates in the following three modes:

1. Vector mode - the vector mode is one in which a line is drawn from one location to another. It has eleven binary bit resolution in both the X and Y directions. Thus, 2^{11} or 2048 units make full scale on the eighteen by thirteen inch area mentioned above. These units are referred to as raster units. In this display unit, the (0,0) coordinates correspond to the lower left corner, and the X and Y are measured positive to the right and up respectively. While in the vector mode, the speed of drawing vectors is 0.22 inches per microsecond.
2. Point mode - the point mode also has eleven binary bit resolution. A point can be plotted on one edge of the display area and a jump to the other edge of the display area takes approximately forty-four microseconds. A half screen jump takes about twenty-three microseconds.

3. Character mode - the character mode is capable of producing any one of sixty-four characters. The characters may be displayed in any one of three sizes: one-eighth inch, one-fourth inch, or three-eighths inches high. The average time to draw a character is five microseconds. If the characters are written in a horizontal line, indexing to the next character is automatic. Appendix C contains a list of the allowable characters for the display unit character generator. It should be noted that the character codes are not the same as SDS internal character codes and each character must be translated before it can be displayed.

The display unit contains two registers: (1.) a command register, and (2.) a string register. The command register is a fifteen bit register which, when the display unit is energized, contains the address of the first cell of the first data string. The string register contains a pointer (command word) to the first word of the string currently being displayed.

The data strings fall into two categories: (1.) a vector or point mode, and (2.) a character mode. In each case, the first word of the string is the same. It contains and X and Y coordinate at which to position the beam. (See Figure 1 for the computer word format.)

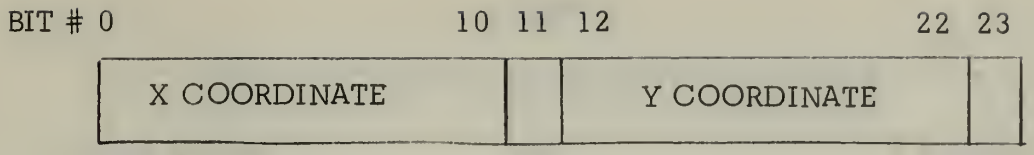


FIGURE 1

FIRST WORD FORMAT (ANY MODE)

The second word in the string will specify whether the vector, point, or character mode, the intensity, and, if character, the size of the character. (See Figure 2 for the computer word format and Table 1 for the octal interpretation of the second word.)

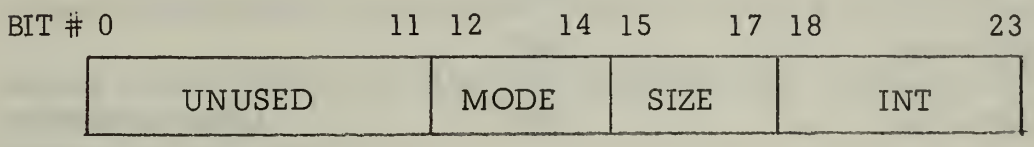


FIGURE 2

SECOND WORD FORMAT

TABLE 1

OCTAL INTERPRETATION OF SECOND WORD

MODE = 0	SET	POINT MODE
MODE = 1	SET	VECTOR MODE
MODE = 2	SET	CHARACTER MODE
SIZE = 0	SET	SMALL SIZE
SIZE = 1	SET	MEDIUM SIZE
SIZE = 2	SET	LARGE SIZE
INT = 00	SET	LEAST INTENSITY
INT = 01	SET	
INT = 02	SET	
INT = 04	SET	
INT = 08	SET	MOST INTENSITY

The third word in the data string of the vector and point mode is the same as the first word. It is the starting point. Each succeeding word in the data string follows the same format. (See Figure 3 for the computer word format.)

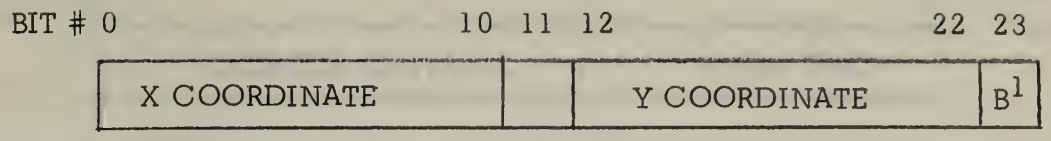


FIGURE 3

THIRD WORD FORMAT

[1]If operating in vector mode and BIT 23 = 1, the display will be directed to the X and Y coordinates as in the point mode (No vector will be drawn). This allows the drawing of disconnected vectors.

In the character mode, the first word specifies the position of the first character; therefore, the third and following words in the data string are characters to be plotted. The characters are packed into the memory cell as four six bit ASC II characters. (See Figure 4 for the computer word format.)

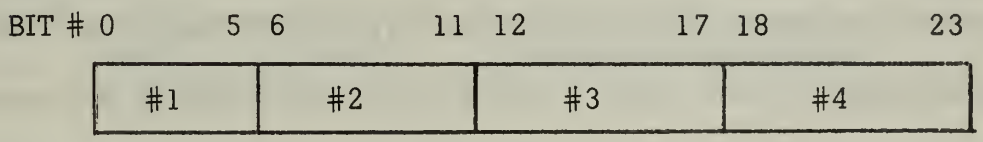


FIGURE 4

CHARACTER MODE DATA WORD[1]

[1]The characters are displayed in the order indicated above (1, 2, 3, 4).

Each data string in memory is represented by a command word. The command word contains the word count (length of the data string) and the address in memory of the first word of the data string. (See Figure 5 for the computer word format.)

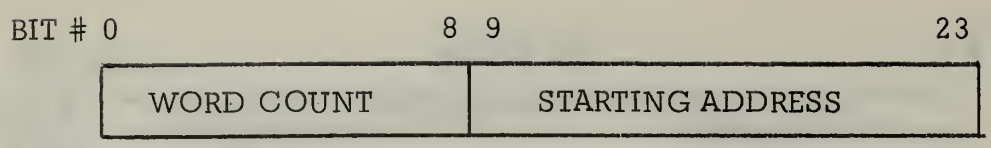


FIGURE 5

COMMAND WORD FORMAT

Data Channels

The analog computer and display unit are interfaced to the digital computer by high speed data channels. Control logic and signal buffering are contained within the high speed data channels. The interface units control data transfer directly to or from a memory location at memory cycle speeds with a minimum of central processor usage.

The display console high speed data channel is initialized with an Energize Output Mode (EOM) instruction followed by the transmission of a command word. As mentioned above, the command word contains the starting address of the first data string, and the number of words which make up the data string. The display then accesses the memory to obtain the data at a rate determined by the interface. The interface decreases the word count by one each time a data word is obtained from the memory. When the word count equals zero, the next word in memory is taken as a new command word containing the starting address and word count of the next data string. If the command word is all zeros, chaining is ended and an interrupt is generated. In the event that both the central processing unit (CPU) and the interface are trying to access

the same bank of memory at the same time, they steal cycles, that is, they take turns in accessing the bank of memory.

Display Console Light Pen and Keyboard

Each display console has two interrupts in addition to the command-word-equals-zero which was mentioned above:

1. Light pen interrupt - the light pen, if enabled, will cause an interrupt if it senses light on the viewing area. At the time of the interrupt, the address of the cell in memory which contained the coordinates of the point the light pen sensed may be read by an EOM instruction. If the light pen pointed to a character, its position in the memory word is also present. Figure 6 demonstrates this word format.

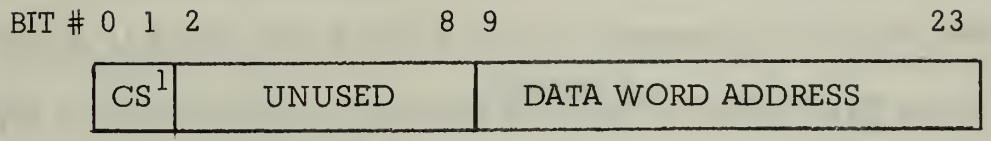


FIGURE 6

FORMAT OF WORD READ BY EOM

[1]CS is 0, 1, 2, or 3 depending on which of the characters of the word was pointed to.

2. Keyboard/function switch interrupt - the depression of a function switch of keyboard key will cause an interrupt, if enabled. A word may then be read into memory by an EOM command. Figure 7 demonstrates this word format.

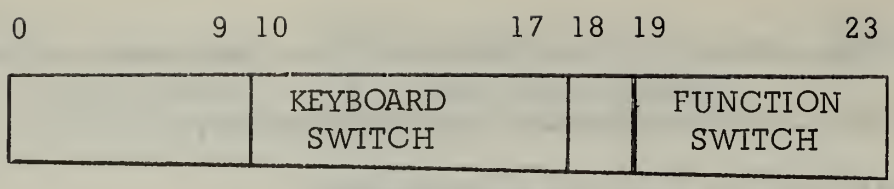


FIGURE 7

FORMAT OF WORD READ BY EOM[1]

[1]Bits 10 - 17 are the ASC II code denoting which key of the keyboard was depressed.
Bit 18 - if 0, indicates a function switch generated the interrupt. If 1, indicates a keyboard key was depressed.
Bits 19 - 23 - a five bit binary number representing which function was depressed.

Appendix B contains a list of interrupt locations.

The individual data strings in memory are connected by command words into a continuous refresh loop. Thus, at the end of each data string is the command word of the following data string. The display unit will chain itself along the refresh loop until: (1.) a command word is zero, or (2.) the command word of the first data string is encountered.

If the first method is used, an interrupt is generated when the command word equals zero. The display is re-energized on a clock pulse. This must be done at least sixty times per second or noticeable flick of the display will be seen.

The second method of refreshing the display forms an endless loop of the data strings. The command word at the end of the last data string contains the starting address and word count of the first data string. Figure 8 shows a series of data strings chained into an endless loop.

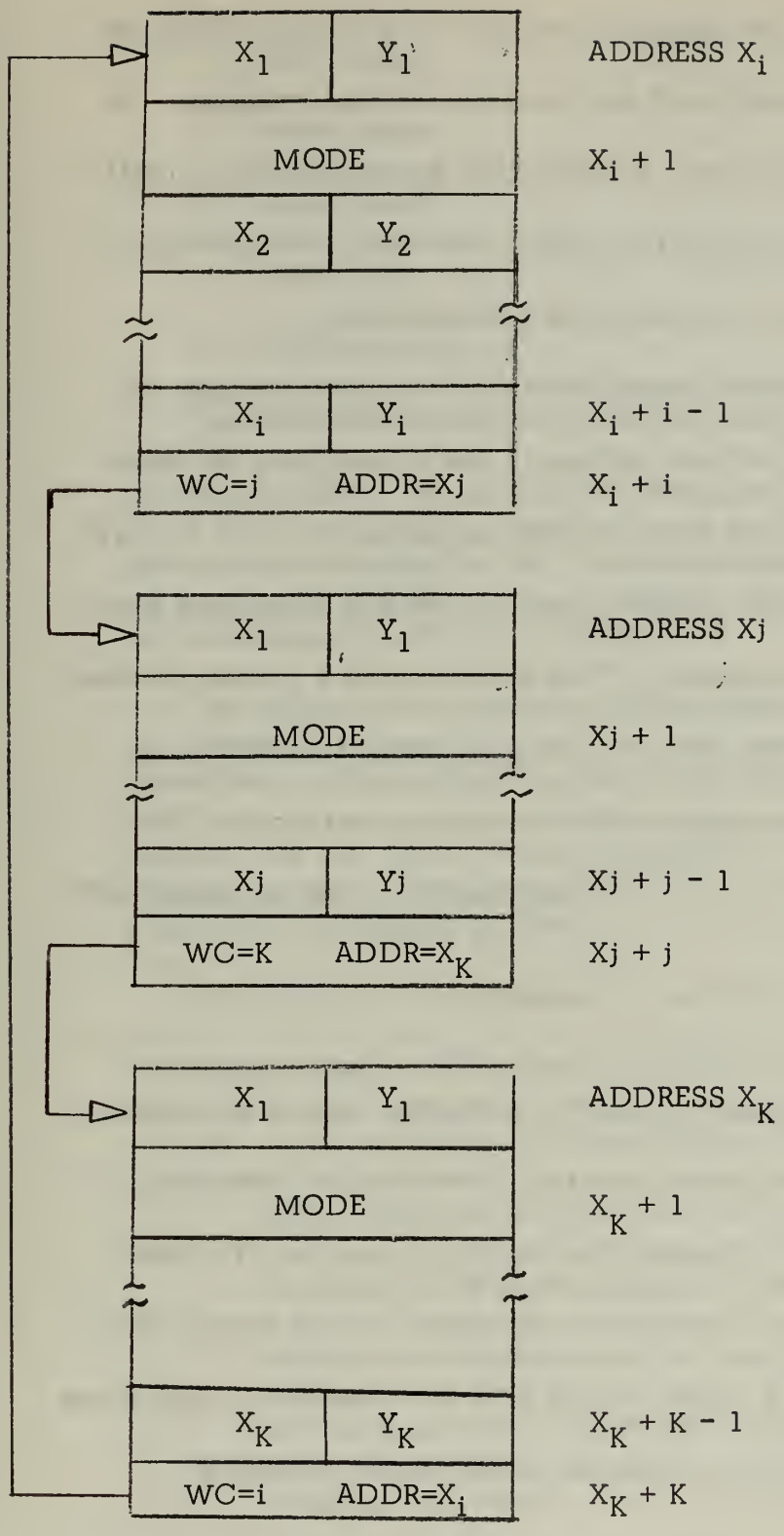


FIGURE 8

CONTINUOUS REFRESH LOOP

The interrupt method is efficient for short refresh loops since the display is idle during a portion of each one-sixtieth of a second. As the refresh loop becomes longer, the idle time becomes smaller until the display of the entire refresh loop takes more than one-sixtieth of a second. If this happens, the result is unpredictable.

The endless loop method insures that all of the data strings are displayed. If the number of data strings in the refresh loop becomes large, there will be a flicker in the display presentation. For a small number of data strings in the refresh loop, the data is displayed more often than sixty times per second. This necessitates a greater number of memory accesses for data than the interrupt method, resulting in more cycle stealing and a slower effective cycle speed for the CPU.

The endless loop method will be implemented in the proposed software package.

Computer Environment

The computer environment in which the display unit is to operate is a hybrid computer. The digital portion is the SDS 930 computer. Appendix A contains the instruction list for this computer. The core memory consists of 16,384 twenty-four bit words with an access time of 0.7 microseconds and a cycle time of 1.75 microseconds. Below are listed the peripheral devices connected to the digital hardware:

1. Typewriter
2. Magnetic tapes - 2
3. Rapid access disc

4. Line printer

5. Photo reader

6. Paper punch

7. Card reader

8. Display units - 2

The Comcor CI 5000 is a medium size general purpose hybrid/ analog computer. At present the computer has 36 operational amplifiers. This can be expanded to 90. There are also resolvers, comparators, and associated hardware available.

The modes of the analog computer (reset, hold, compute) can be controlled by the digital computer. The digital computer can directly address and set ratios on potentiometers.

Computer Environment Software

The display software package operates under the SDS REAL-TIME MONITOR (RTM). RTM is made up of the following elements:

1. Resident monitor - the resident acts as an executive for the other elements of RTM, and an interrupt monitor which saves registers prior to servicing any interrupt. The resident monitor also determines if a subroutine is being re-entered as the result of an interrupt and, if so, saves the previous entering arguments.

2. SDS Fortran IV - the SDS Fortran IV is the standard Fortran IV mathematical language which operates in real time and produces a binary object version of the user's program.

3. SDS Symbol Assembler - the symbol assembler produces the binary object version of an assembly language program.
4. Overlay Loader - the overlay loader supervises the loading of programs which are loaded in segments due to their size.
5. System Input/Output (I/O) Processor - the I/O processor is a general package used to process all input and output operations.
6. Primary Library - the primary library contains mathematical handlers as well as certain routines for debugging programs.

The Interrupt System

An interrupt is a signal external to the computer main frame which causes the computer to transfer program control to one of a selected set of memory locations. Some of these interrupts are generated by the REAL-TIME CLOCK and by the hybrid hardware. Others are signalled from the analog/hybrid computer via patchable trunklines, or from the display units. When an interrupt wire has a signal (pulse) on it, the computer executes the instruction in the location in memory corresponding to the interrupt wire. There is an hierarchy among the interrupts. The three states of an interrupt are:

1. Inactive - the interrupt does not have a signal on it.
2. Waiting - the interrupt signal has been received, but the computer is presently processing an interrupt of higher priority.
3. Active - the interrupt signal has been received, and the computer is processing the interrupt. When the interrupt processing is

complete, the interrupt returns to the inactive state. An interrupt can be moved from the active state to the waiting state by the arrival of higher priority. When the processing of the higher priority interrupt is completed, the next higher priority waiting interrupt is then moved to the active state.

Programming Interrupts

Interrupts may not always be used for the same function. For example, the light pen may be used to draw a curve in one instance and used to delete a curve in another. The reprogramming of the light pen interrupt is accomplished by the function switch interrupt. The function switches (FS) are assigned specific uses. To continue the example above, let depressing FS_i indicate that the light pen is to draw a curve. Depressing FS_j indicates that the light pen is to delete a curve that is specified.

Assume that cell 204 (the keyboard/function switch interrupt location) contains a branch to a subroutine which interprets which key or switch was depressed.

In operation FS_i is depressed. This executes the instruction in cell 204 which branches to the interpretive subroutine (SUB 1). This subroutine determines that FS_i was depressed and the subroutine to draw a curve (SUB 2) is desired when the light pen interrupt arrives. SUB 1 then places a branch to SUB 2 in cell 205 (the light pen interrupt location). When the light pen interrupt arrives, SUB 2 will be executed.

CHAPTER III

FUNCTIONAL DESCRIPTION OF ELEMENTS

The display unit software package envisioned will contain two major functional groups:

1. Curve plotting
2. Tactical situation plotting

Each of these groups will have elements in common with the other group, yet by breaking the software package down in this manner, the overall complexity of the package is greatly reduced. Examining the first group in more detail, the following elements are found necessary:

1. CENTER - Center determines a bias to be added to all X and Y coordinates so that data will be centered on the display area.
2. SCALE - Scale works in conjunction with Center to insure that the maximum value in problem units does not exceed the 2048 raster units for the display.
3. MULTIPLY - Multiply is an element that multiplies each X and Y coordinate by the scale factor determined by the element Scale and adds the factor determined by the element Center.
4. PACK - Pack assembles the X and Y coordinate into the proper word format for the display unit.
5. AXIS - Axis uses the bias determined by Center, and draws axes on the display area.
6. GRID - Grid draws grids on the face of the display area.

7. GRIDIV - Gridiv generates a vector array of points along the X and Y axes through which lines will be drawn to make the grid, if the user desires to use a particular type of grid.

8. WORD - This element will display alphanumeric words of any length in any position on the display area.

9. ASSEMBLE - When data strings are generated from the above elements, they must be assembled into a continuous loop for refreshing the display.

10. AXIS ARRAY - Axis array generates a vector array of marks to be placed along the X and Y axes to indicate problem units.

11. CURVE CHANGE - Curve change uses the light pen to determine which curve is to be changed and to make the changes to the curve during the display.

12. CURVE DELETE - Curve delete will erase a curve after being specified by the light pen and that particular curve will no longer be displayed.

13. CURVE ADD - Curve add will make the necessary changes to the refresh loop generated by the element Assemble and the new curve will be added to the page currently being displayed.

14. CURVE DRAW - This element will take positions of the light pen as it moves across the display area, make a list of the X and Y coordinates it traces out, and generates these into a curve. (It should be noted, however, that this curve will not be a smooth curve, but will actually connect the points marked by the light

pen; a first or second order approximation or refining of this curve would be another routine.)

Figure 8 below shows a block diagram of the program controlled elements of this group. The lines connecting the blocks indicate paths of information. Figure 9 indicates the information flow between the interrupt controlled elements of this system function.

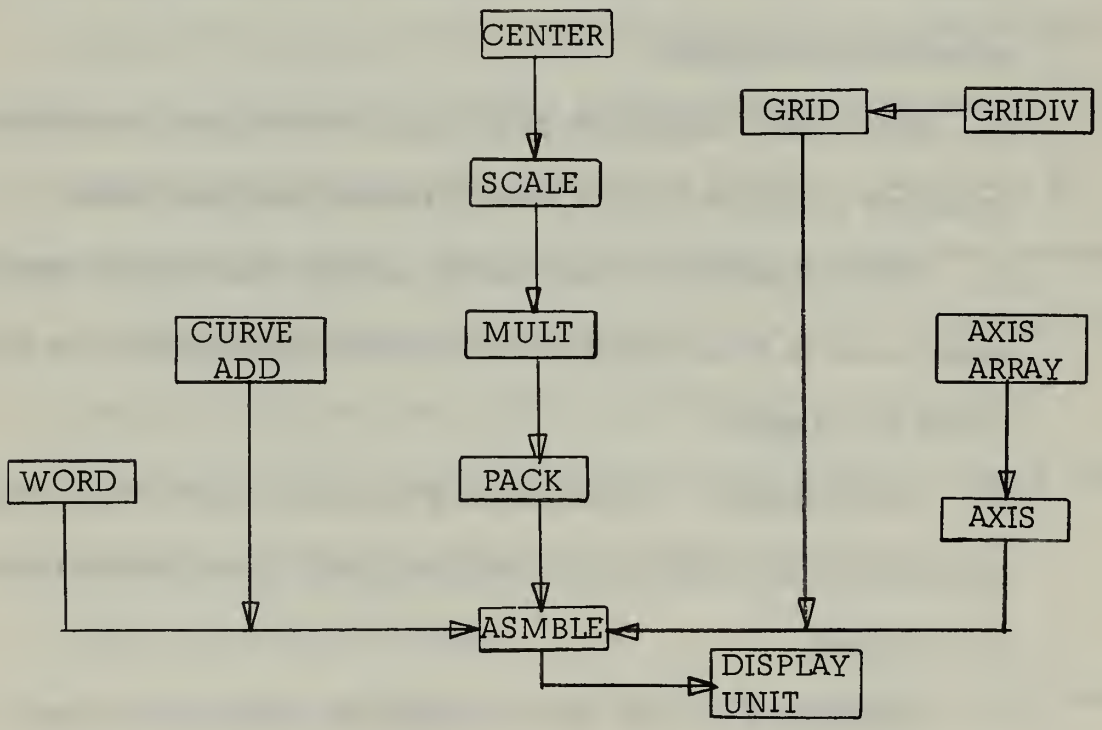


FIGURE 9

PROGRAM CONTROLLED SUBROUTINES

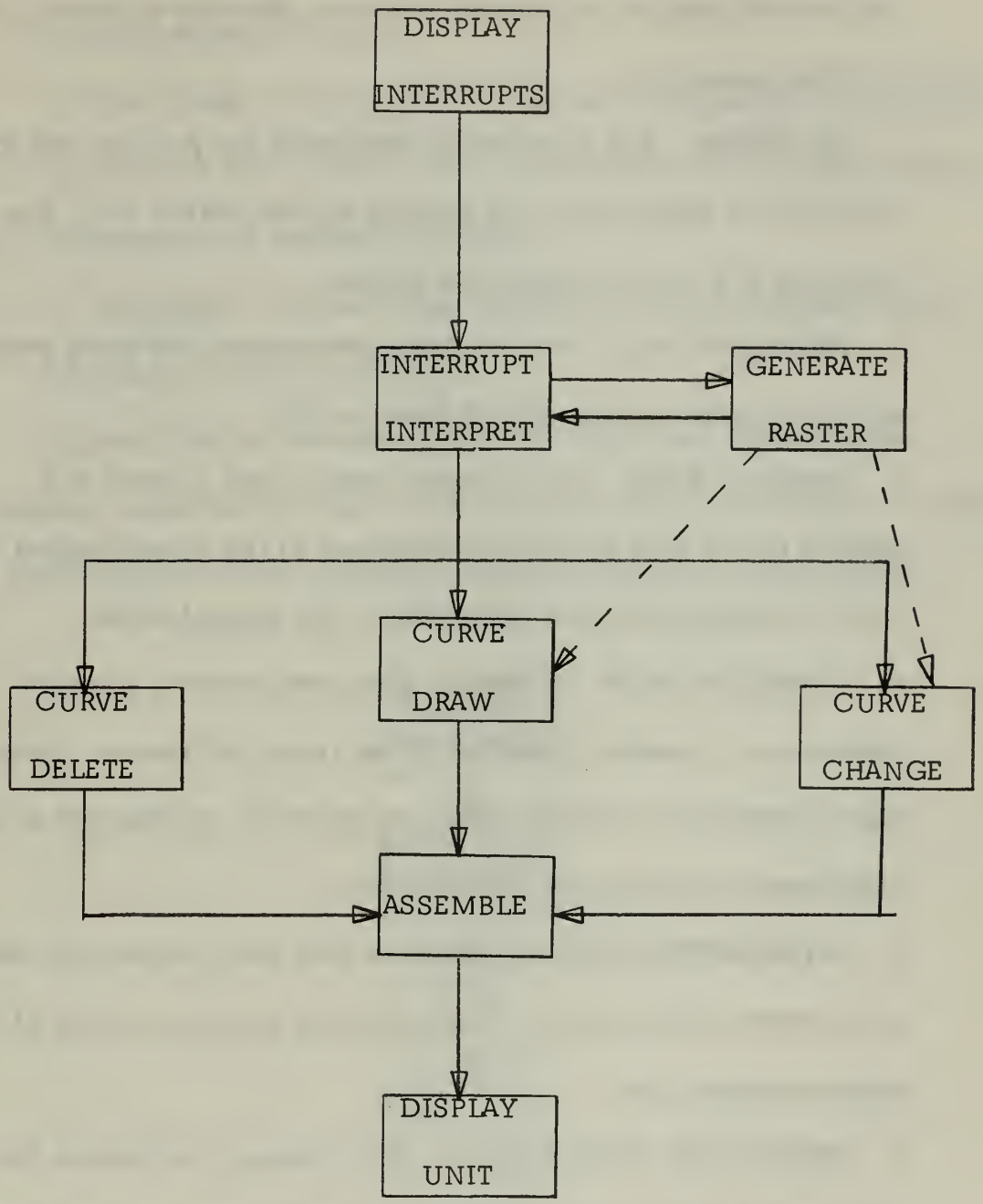


FIGURE 10

INTERRUPT CONTROLLED SUBROUTINES

Tactical Situation Plot

The second group is the tactical situation plot and is made up of the following elements:

1. FIX ORIGIN - Fix origin would determine the point on the display surface from which range and bearing will be marked off. The origin can move if it is on a target that moves.

2. PROBLEM SCALE - Problem scale determines how many problem units will make up a number of raster units.

3. POSITION PLOT - This element breaks down a range and bearing by the sine and cosine functions to the corresponding X and Y coordinates, and adds them to the present origin.

4. CHARACTER PLOT - Character plot plots either a standard character or a special character at the range and bearing specified. These characters represent ships or airplanes, or whatever is being displayed on the tactical situation plot.

5. INFORMATION PLOT - Information plot plots course and speed at a certain fixed point in reference to the present position of a target situation plot.

6. SPECIFY PAST INFORMATION - This element determines how many past positions will be drawn in a track mode. It will be a plot of points back from the present of the target. The amount of past information to be plotted becomes a function of the available core space and number of targets plotted.

7. UNIT DELETE - This element will delete all information corresponding to a target.

8. UNIT ADD - A track will be started at the position designated by the light pen. Information corresponding to this target will be generated as it becomes available.

9. ASSEMBLE - This element assembles the data strings into a continuous loop for refreshing.

Figure 10 below indicates the information flow between the interrupt controlled elements of this system function. Figure 11 shows a block diagram of the program controlled elements of this group.

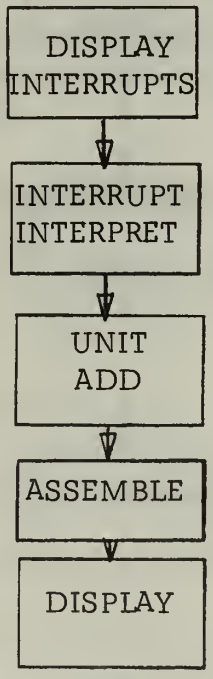


FIGURE 11

INTERRUPT CONTROLLED SUBROUTINES

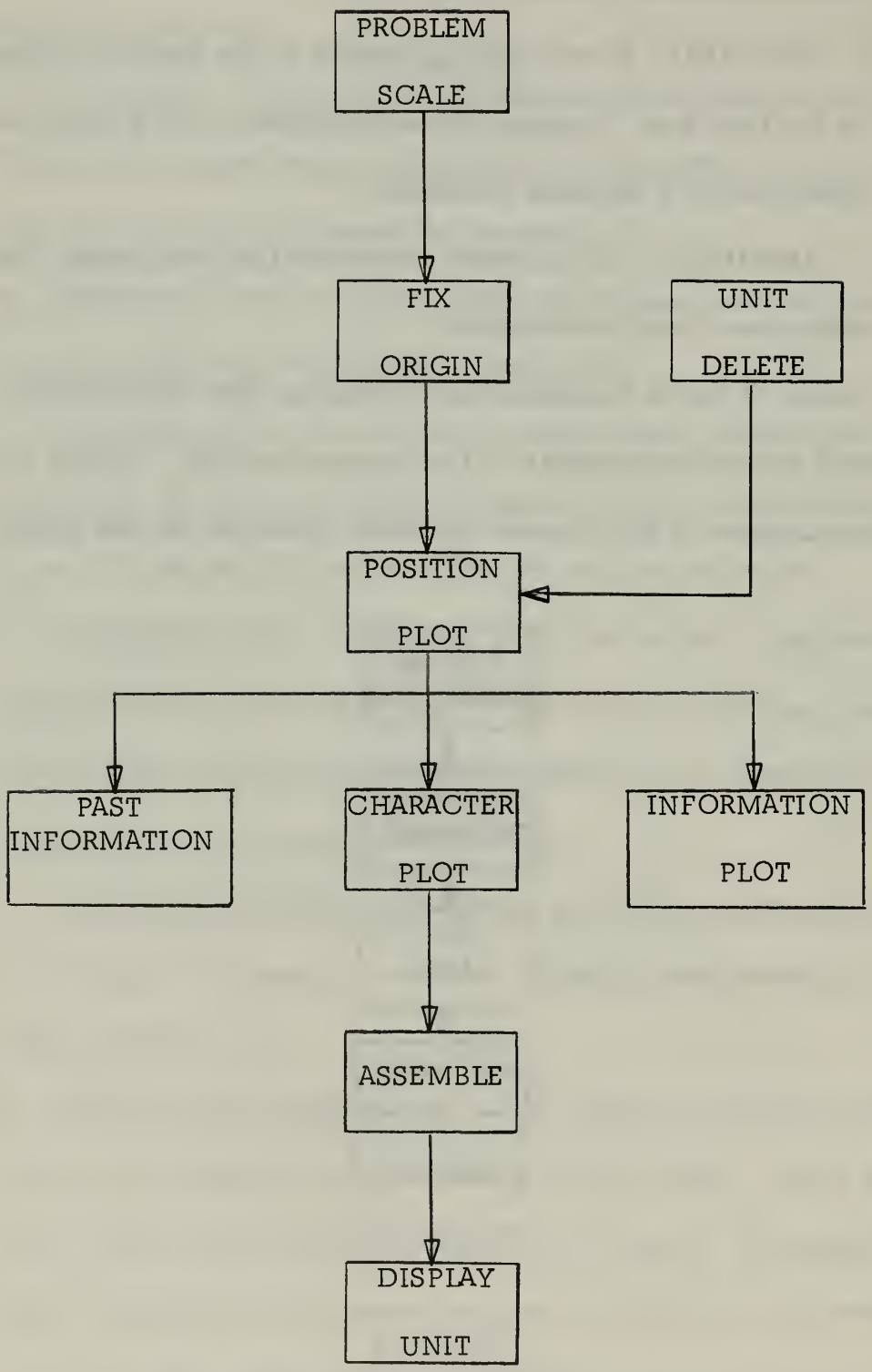


FIGURE 12

PROGRAM CONTROLLED SUBROUTINES

Functional Description of Elements

Functional Subset Draw

1. CENTER - This subroutine computes X and Y bias values to fix the smallest X and Y values to the bottom and left margins respectively. Subroutine Center is called in the following manner:

CALL CENTER (X, Y, NPTS, ILM, IRM, ITM, IBM, XBIAS, YBIAS)

Description of parameters:

X - An array of length NPTS of X coordinates to be plotted. (Single precision floating point values.)

Y - An array of length NPTS of Y coordinates to be plotted. (Single precision floating point values.)

NPTS - An integer value corresponding to the length of the X and Y arrays.

ILM - Left margin in raster units.

IRM - Right margin in raster units.

ITM - Top margin in raster units.

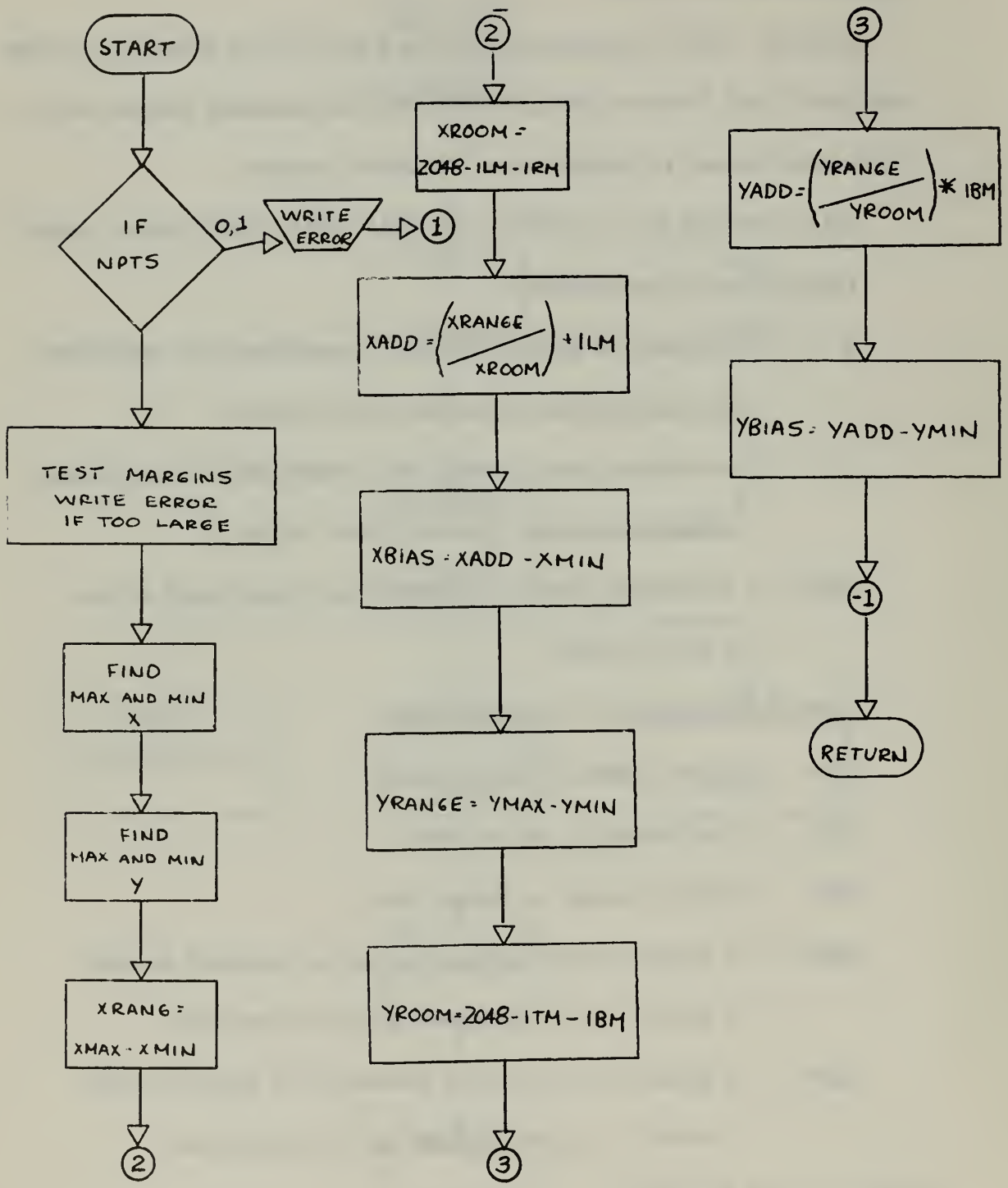
IBM - Bottom margin in raster units.

XBIAS - A problem unit floating number to be added to each X value. It is calculated by the subroutine.

YBIAS - A problem unit floating number to be added to each Y value. It is calculated by the subroutine.

Discussion of flowchart:

Flowchart 1 indicates how subroutine Center calculates XBIAS and YBIAS. Upon entry, the number of points NPTS are tested to insure



FLOWCHART 1

SUBROUTINE CENTER

that is is positive. The margins are tested to see if there is sufficient display area left to plot the curves. Next, the maximum and minimum values of X and Y are found. From these, XADD is calculated which corresponds to a floating point problem unit number which represents the left margin (ILM). XBIAS then equals the minimum value of X subtracted from XADD. The same process is used to obtain XBIAS.

2. SCALE - This subroutine computes floating point multipliers to multiply the X and Y arrays. Subroutine scale is called in the following manner:

CALL SCALE (X, Y, NPTS, XBIAS, YBIAS, ILM, IRM, ITM, IBM, XSC, YSC, IXZERO, IYZERO)

Description of parameters:

X - An array of length NPTS of X coordinates to be plotted. (Single precision floating point values.)

Y - An array of length NPTS of X coordinates to be plotted. (Single precision floating point values.)

NPTS - Length of the X and Y arrays.

XBIAS - A floating point value to be added to each X value.

YBIAS - A floating point value to be added to each Y value.

ILM - Left margin in raster units.

IRM - Right margin in raster units.

ITM - Top margin in raster units.

IBM - Bottom margin in raster units.

XSC - A floating point number to multiply each X value.

YSC - A floating point number to multiply each Y value.

IXZERO - A raster unit number which corresponds to the zero point of the X array.

IYZERO - A raster unit number which corresponds to the zero point of the Y array.

Discussion of Flowgraph:

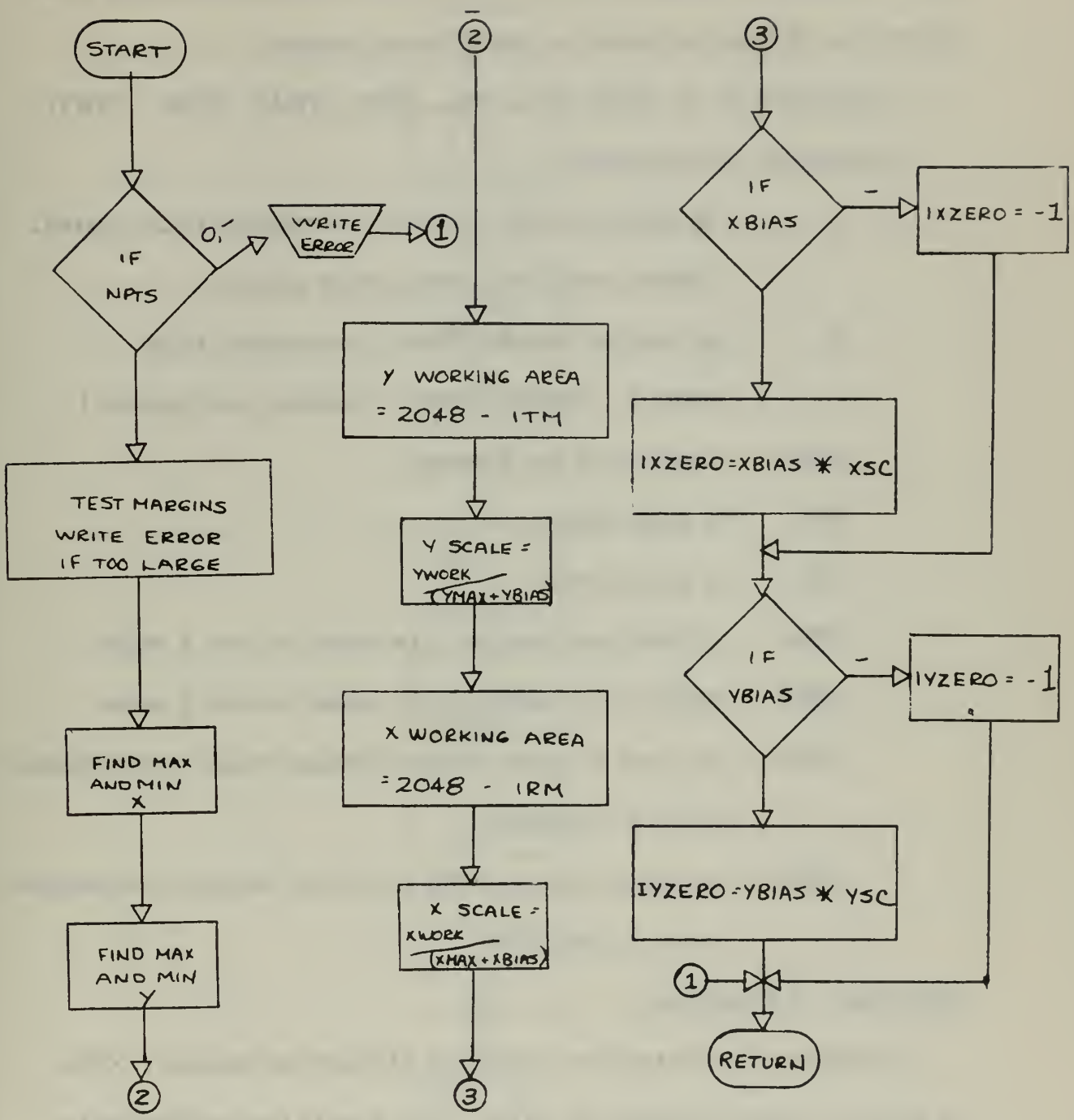
Figure 2 indicates how subroutine Scale progresses. NPTS, or the length of the X and Y arrays, is checked to insure that it is positive. The margins are then tested to insure that there is sufficient area on the display area to display the curves. Maximum and minumum values of X and Y are found, and the working area in both the X and Y directions are found. Since subroutine Center calculated a constant to insure that the smallest values of X and Y were on the display area, subroutine Scale insures that the largest values of X and Y are on the display area. The maximum integer value that X can be is 2048 minus the right margin. This corresponds to the maximum value of X plus the XBIAS. The X scale factor is then:

$$XSC = (2048 - IRM) \div (XMAX + XBIAS)$$

If XBIAS is negative, IXZERO is set to -1. If it is positive, IXZERO equals the integer value of XBIAS times XSC.

If IXZERO or IYZERO are off the display area, they are set to -1. This will act as a flag to subroutine AXIS to inform it that the axis cannot be drawn.

The same procedure is followed for the Y values and the Y scale factor (YSC) and the scaled zero point (IYZERO) are found.



FLOWCHART 2

SUBROUTINE SCALE

3. MULTIPLY - This subroutine takes the X and Y arrays and converts them to integer values scaled so that all points are on the display area. Subroutine Multiply is called in the following manner:

CALL MULT (X, Y, NPTS, XSC, YSC, XBIAS, YBIAS, NEWX, NEWY)

Description of parameters:

X - An array of length NPTS of X coordinate to be plotted. (Single precision floating point values.)

Y - An array of length NPTS of Y coordinate to be plotted. (Single precision floating point values.)

NPTS - Length of X and Y arrays.

XSC - X scale factor.

YSC - Y scale factor.

XBIAS - Problem unit number to be added to each X value.

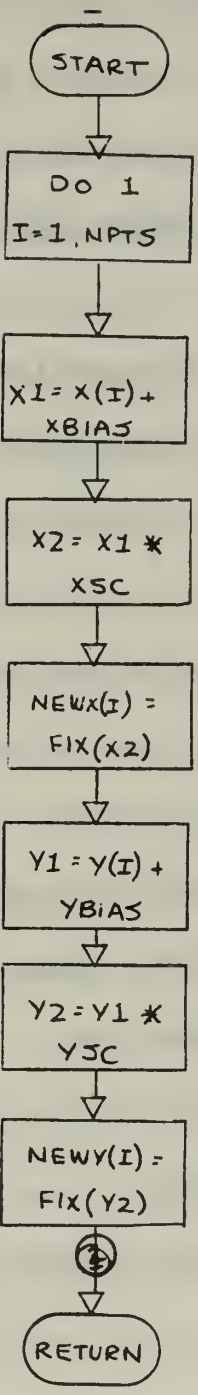
YBIAS - Problem unit number to be added to each Y value.

NEWX - An array of length NPTS of integer values corresponding to the X coordinate.

NEWY - An array of length NPTS of integer values corresponding to the Y coordinate.

Discussion of Flowgraph:

Flowgraph 3 indicates how subroutine Multiply progresses. XBIAS is added to each X problem unit value. This sum is then multiplied by XSC to obtain a floating point scaled value. The integer value of this number is stored in the NEWX array. The same procedure is followed for each Y value.



FLOWCHART 3

SUBROUTINE MULTIPLY

4. PACK - This subroutine takes the arrays NEWX and NEWY and packs the corresponding elements of each array into the array IPLOT. The first two words mentioned in the Chapter II are the special code words. Subroutine PACK is called in the following manner:

CALL PACK (NEWX, NEWY, MODE, ISIZE, INT, IPLOT, ICODE)

Description of parameters:

NEWX - Integer array of length NPTS of scaled X coordinates.

NEWY - Integer array of length NPTS of scaled Y coordinates.

MODE - If MODE = 0, a display of unconnected points is generated.

If MODE = 1, a curve of connected points is generated.

ISIZE - Size of points or curves.

ISIZE = 0 small size

ISIZE = 1 medium size

ISIZE = 2 large size

INT - Intensity of points or curves.

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

INT = 08 most intensity

IPLOT - An array of NPTS +3 which is the curve data string. This array must be dimensioned in the user's main program.

ICODE - A code word containing the length of the array IPLOT and the starting address of the curve data string.

Discussion of Flowchart:

The integer scaled NEWX is shifted left thirteen places and the integer scaled NEWY is shifted left one place. These values are packed into the corresponding cell of IPLOT. This is the form required by the display unit. The first word in the array IPLOT is the same as the third word. The second word of IPLOT contains the mode, size, and intensity code.

5. AXIS - This subroutine draws a horizontal line (X axis) from the left margin to the right margin through IYUP. A vertical line (Y axis) is drawn from the bottom margin to the top margin through the point IXOVR. Subroutine AXIS is called in the following manner:

CALL AXIS (IXOVR, IYUP, ILM, IRM, IBM, ITM, ISTNG, ICODE)

Description of parameters:

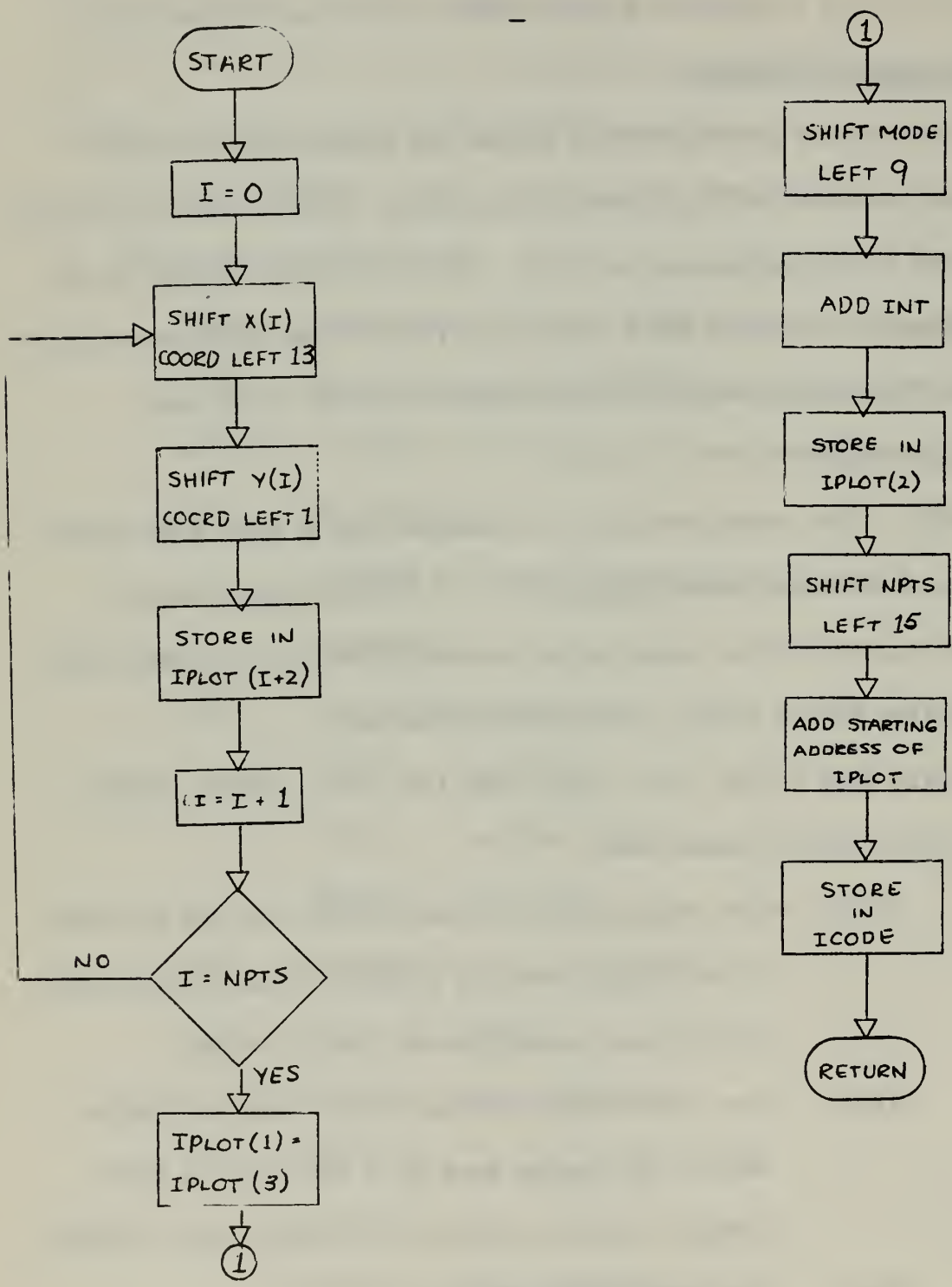
IXOVR - The integer number of raster units from the left side of the display area the Y axis is to be drawn through. Under usual circumstances, this is IXZERO.

IYUP - The integer number of raster units from the bottom side of the display area the X axis is to be drawn through. Under usual circumstances, this is IYZERO.

ILM - Left margin in raster units.

IRM - Right margin in raster units.

ITM - Top margin in raster units.



FLOWCHART 4

SUBROUTINE P CK

IBM - Bottom margin in raster units.

ISTNG - The axis data string which must be dimensioned in the user's program. The size of dimension is 7.

ICODE - A code word containing the length of the array ISTNG and the starting address of the axis data string.

Discussion of flowchart:

The values of IXOVR and IYUP are tested to insure that they fall inside the assigned margins. If either or both do not fall on the viewing area, messages stating this are typed out. If the X axis is to be drawn, the X coordinates are ILM and 2048 - IRM. The Y coordinate for these points is IYUP. The trace is then blanked from the end of the X axis to the start of the Y axis. The Y coordinate of the Y axis are IBM and 2048 - ITM. The X coordinate for the Y axis is IXOVR. The intensity is set to medium.

Figure 13 below indicates how these values actually work:

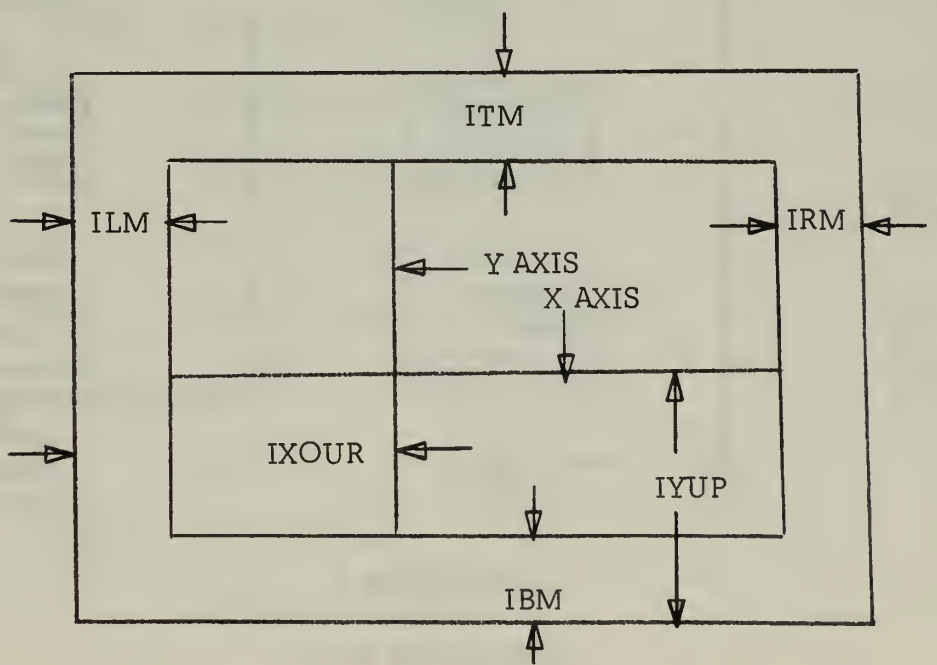
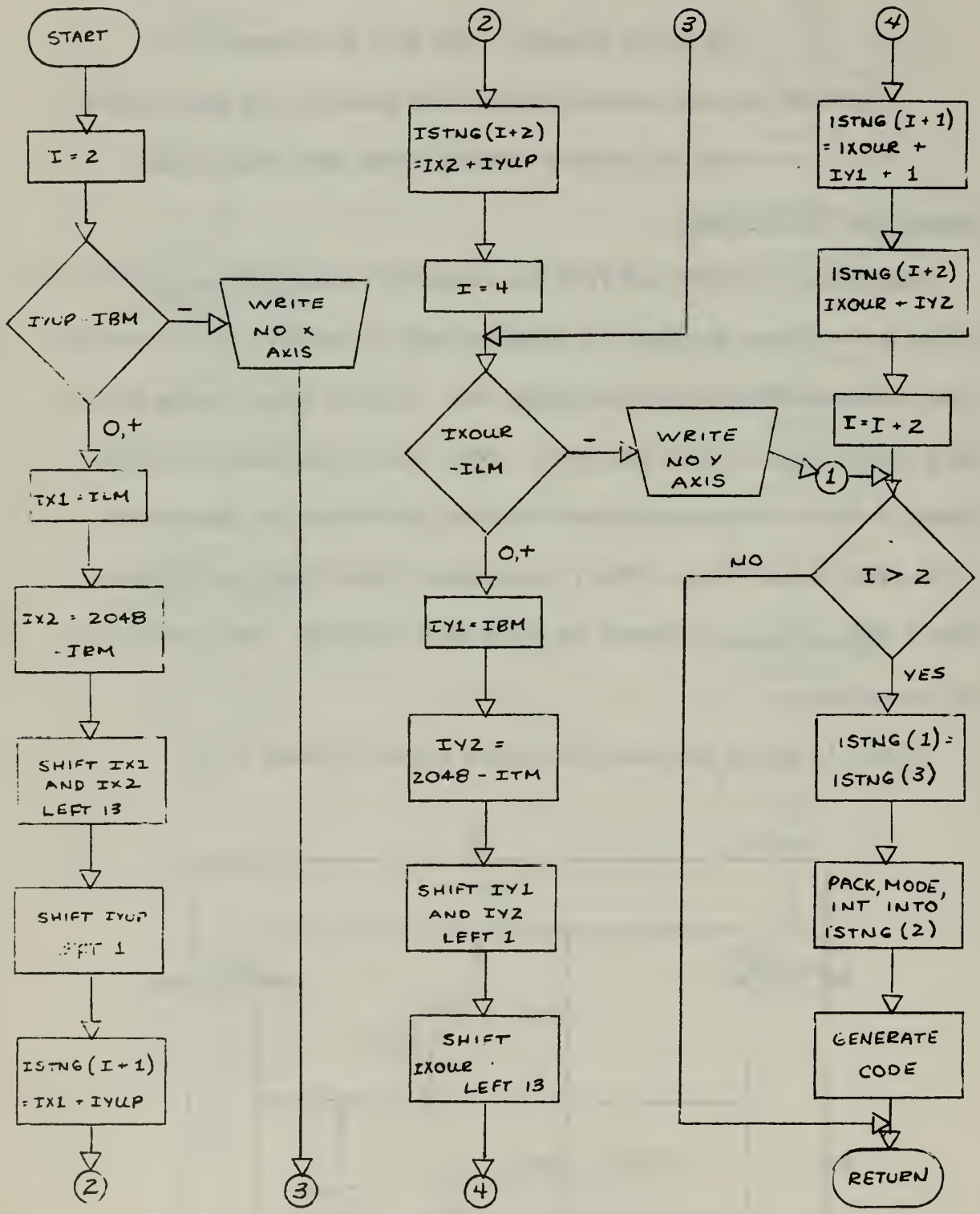


FIGURE 13

DISPLAY VIEWING AREA



FLOWCHART 5

SUBROUTINE AXIS

6. GRID - This subroutine draws vertical lines through the points of the array IXPT. These lines go from the bottom margin to the top margin. Horizontal lines are drawn through the points of the array IYPT. These lines go from the left margin to the right margin.

Note: Subroutine GRIDIV will generate the arrays IXPT and IYPT in one of three modes. These will be discussed under GRIDIV.

Subroutine GRID is called in the following manner:

CALL GRID (IXPT, IYPT, IXNPT, IYNPT, ILM, IRM, ITM, IBM, MODE, ISIZE, INT, IGRID, IGRD)

Description of parameters:

IXPT - An integer array of length IXNPT of X positions through which vertical lines will be drawn.

IYPT - An integer array of length IYNPT of Y positions through which horizontal lines will be drawn.

IXNPT - Length of array IXPT.

IYNPT - Length of array IYPT.

ILM - Left margin in raster units.

IRM - Right margin in raster units.

ITM - Top margin in raster units.

IBM - Bottom margin in raster units.

MODE - MODE = 0 a grid of points

MODE = 1 a grid of lines

ISIZE - Size of lines

ISIZE = 0 small

ISIZE = 1 medium

ISIZE = 2 large

INT - Intensity of trace

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

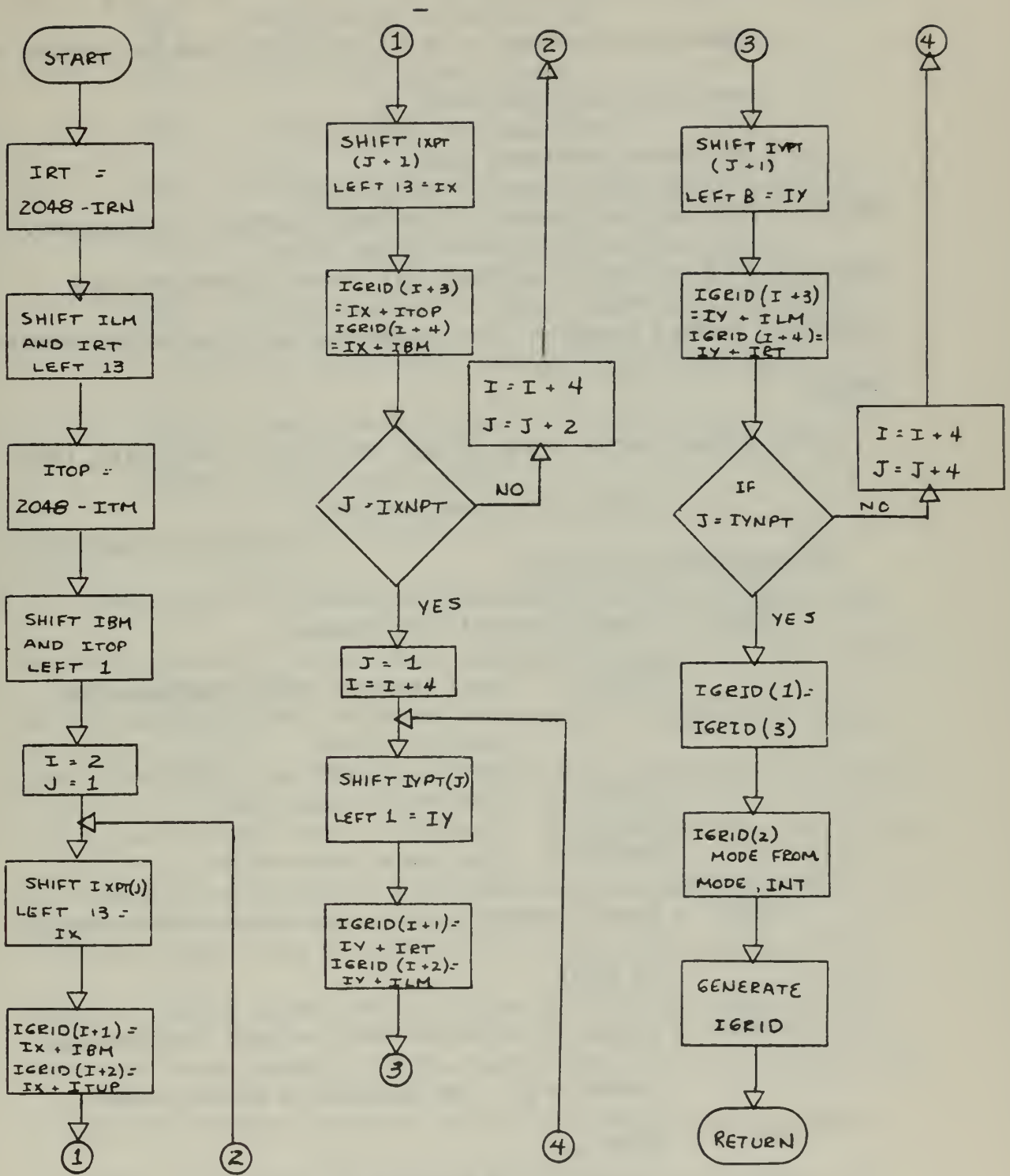
INT = 08 most intensity

IGRID - The grid data string must be dimensioned in the user's program (dimension: 90).

ICODE - A code word containing the length of the data string IGRID and the first address of the string.

Discussion of flowchart:

The X coordinate for each point through which lines are drawn is shifted to the proper position in the computer word. The Y coordinate for each point is likewise shifted. The grid starts out at the intersection of the bottom and left margins and a line is drawn to the intersection of the left and top margins. The X coordinate is then moved to the value specified by the appropriate element of the array IXPT. A vertical line is then drawn down to the bottom margin and the process is repeated. Horizontal grid lines are drawn through the points of the array IYPT in the same manner. The second word of the data string is generated with the user's values of mode, size, and intensity. The code word (ICODE) is generated with the length of the array and the starting address.



FLOWCHART 6

SUBROUTINE GRID

7. GRIDIV - This subroutine generates X and Y arrays in one of three types. These are used by subroutine GRID. Subroutine will generate:

1. A linear (even spacing) with the user specifying the spacing, or
2. A one cycle logarithmic grid, or
3. A two cycle logarithmic grid.

The grid types may be different for the X and Y arrays. For example, there may be a two cycle logarithmic grid in the X direction and a linear grid in the Y direction. This subroutine is called in the following manner:

CALL GRIDIV (IXTYP, IYTYP, IXDIV, IYDIV, ILM, IRM, ITM, IBM, IXPT, IYPT, IXNPT, IYNPT)

Description of parameters:

IXTYP - Type of vertical lines desired.

IXTYP = 0 even spacing, IXDIV determines the number of lines.

IXTYP = 1 1log scale - one cycle

IXTYP = 2 2log scale - two cycles

IYTYP - Type of horizontal lines, the codes are the same as for IXTYP.

IXDIV - Number of horizontal lines desired. The maximum number is 20. The argument is ignored unless IYTYP = 0.

IYDIV - Number of horizontal lines desired. The maximum number is 20. The argument is ignored unless IYTYP = 0.

ILM - Left margin in raster units.

IRM - Right margin in raster units.

ITM - Top margin in raster units.

IBM - Bottom margin in raster units.

IXPT - An integer array of length IXNPT of X coordinates through which vertical lines will be drawn.

IYPT - An integer array of length IYNPT of Y coordinates through which horizontal lines will be drawn.

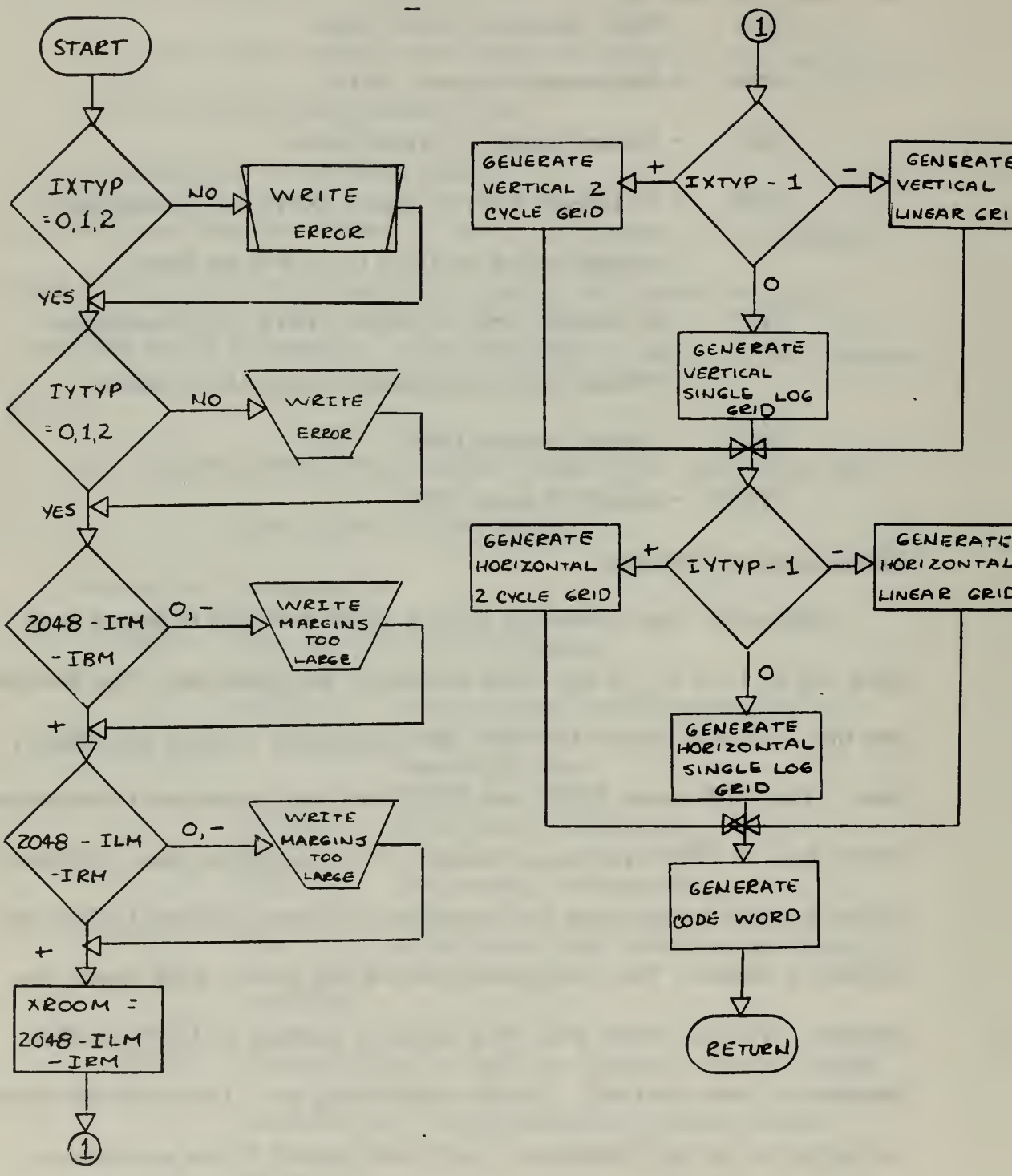
IXNPT - Length of array IXPT.

IYNPT - Length of array IYPT.

Discussion of flowchart:

Upon entry, the subroutine checks IXTYP and ITYP to insure that they are 0, 1, or 2. If not, error messages are typed out. The margins are then tested to insure that they have sufficient room on the display area. The GRID types (IXTYP and IYTYP) are then inspected to determine which type of GRID spacing is desired. The subroutine then will branch to the appropriate location and calculate the array IXPT and IYPT in the following manner: The space available for the grid is 2048 minus the margins. For the linear grid, this space is divided by IXDIV or IYDIV (number of lines desired). For the logarithmic grid, the available space is multiplied by the logarithmic coefficient stored in the subroutine.

8. WORD - This subroutine takes a BCD message of any length and displays it in a horizontal line with the left most character at the X and Y positions designated. This subroutine is called in the following manner:



FLOWCHART 7

SUBROUTINE GRIDIV

CALL WORD (IXPOS, IYPOS, LABEL, LNGTH, ISIZE, IN, IWORD,
ICODE)

Description of parameters:

IXPOS - Integer raster unit horizontal position for the left character in LABEL.

IYPOS - Integer raster unit vertical position of line of characters.

LABEL - A BCD string of any length of characters to be plotted. See Appendix C for the list of allowable characters.

LNGTH - The number of COMPUTER WORDS which contain LABEL. The characters must be packed four characters per COMPUTER WORD.

ISIZE - The size of characters to be plotted.

ISIZE = 0 small size

ISIZE = 1 medium size

ISIZE = 2 large size

INT - Intensity of characters

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

INT = 08 most intensity

IWORD - A data string of characters to be plotted.

ICODE - A code word containing the length of the data string IWORD and the starting location of the string.

Discussion of flowchart:

Each word of LABEL is loaded into temporary storage and examined one character at at time. Since the display does not use the BCD form of LABEL, each character is changed to the corresponding ASC II character for the display unit. After the four characters of one word have been changed, the new word is stored in the appropriate location in the array IWORD. After all characters in the array LABEL have been changed, the X and Y coordinates at which the message is to be displayed are packed into the first word of data string IWORD. The second word of the data string is made up of the intensity and size arguments. The code word containing the length of the data string and its starting location is then generated.

9. AXIS ARRAY - This subroutine places dots along the X and Y axis to represent the user's problem units. The dots are ten raster units below the X axis and ten raster units to the left of the Y axis to make them more visible. This subroutine is called in the following manner:

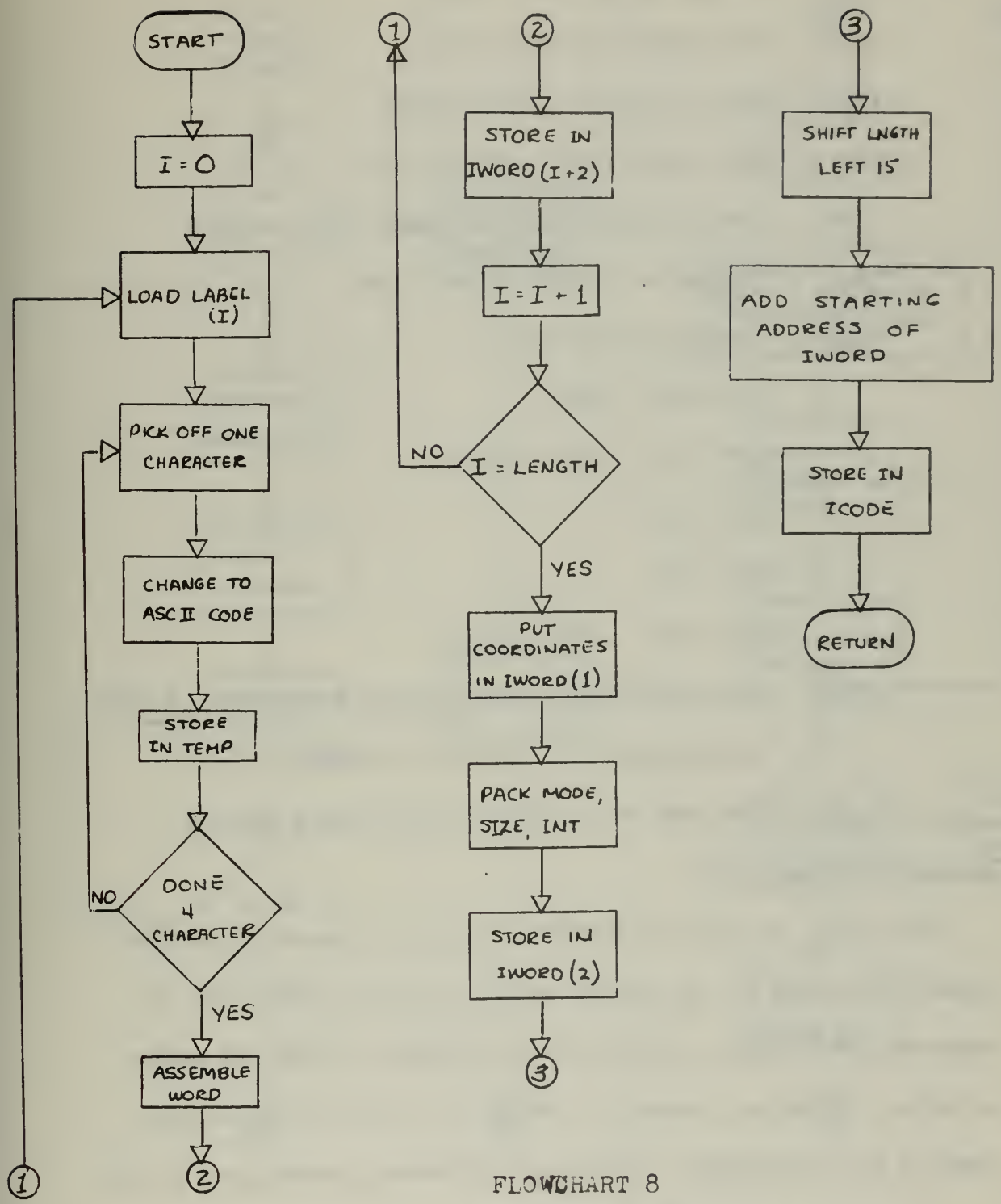
CALL AXARAY (XPU, YPU, IXZERO, IYZERO, XSC, YSC, ILM, IRM, ITM, IBM, INT, ISTNG, ICODE)

Description of parameters:

XPU - Spacing of marks along the X axis in problem units.

YPU - Spacing of marks along the Y axis in problem units.

IXZERO - The X coordinate of the Y axis.



FLOWCHART 8

SUBROUTINE WORD

IYZERO - The Y coordinate of the X axis.

XSC - The X direction scale factor.

YSC - The Y direction scale factor.

ILM - The left margin in raster units.

IRM - The right margin in raster units.

ITM - The top margin in raster units.

IBM - The bottom margin in raster units.

INT - Intensity of the trace.

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

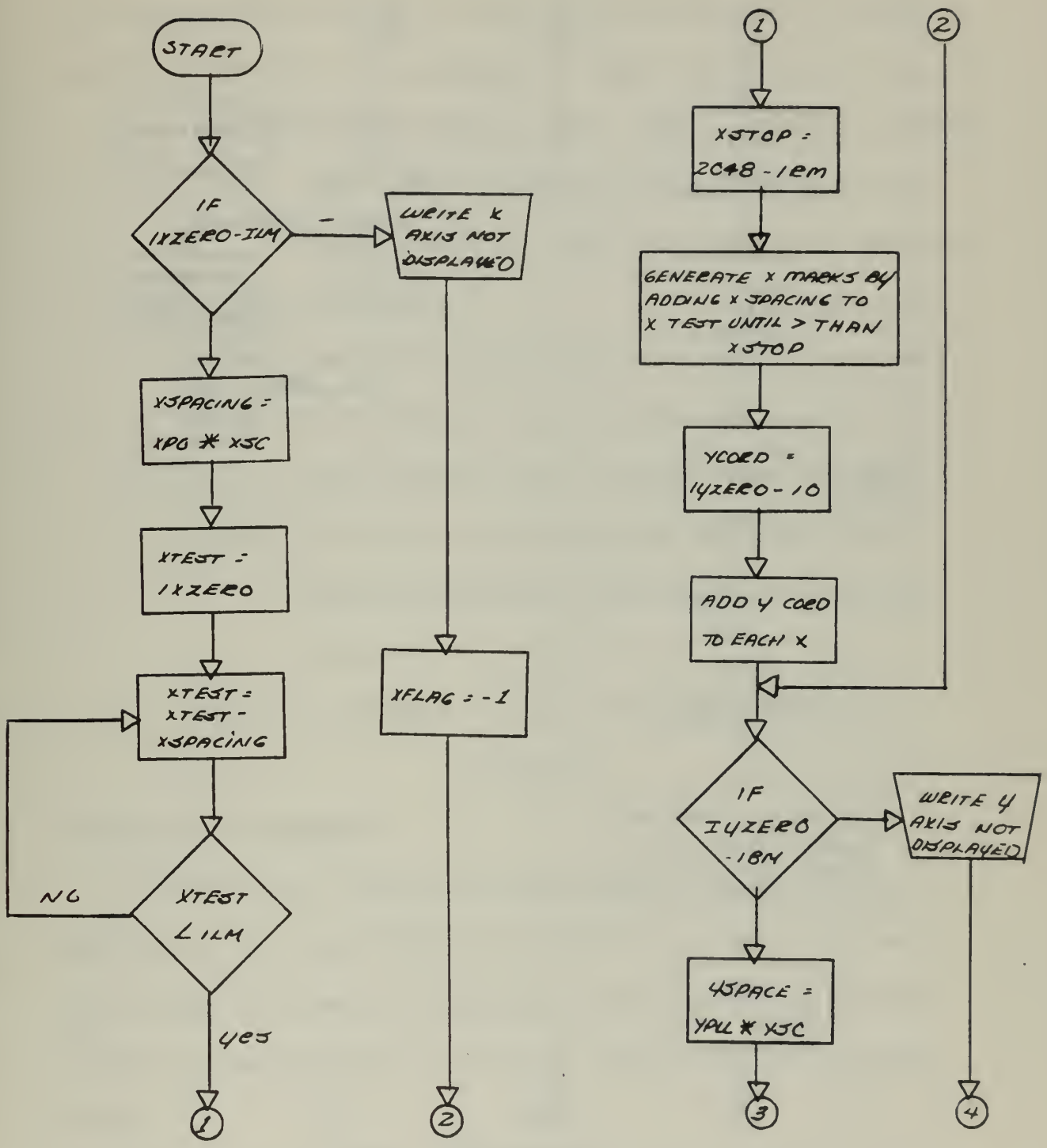
INT = 08 most intensity

ISTNG - The data string generated by the subroutine. It must be dimentioned in the user's program.

ICODE - The code word containing the starting address.

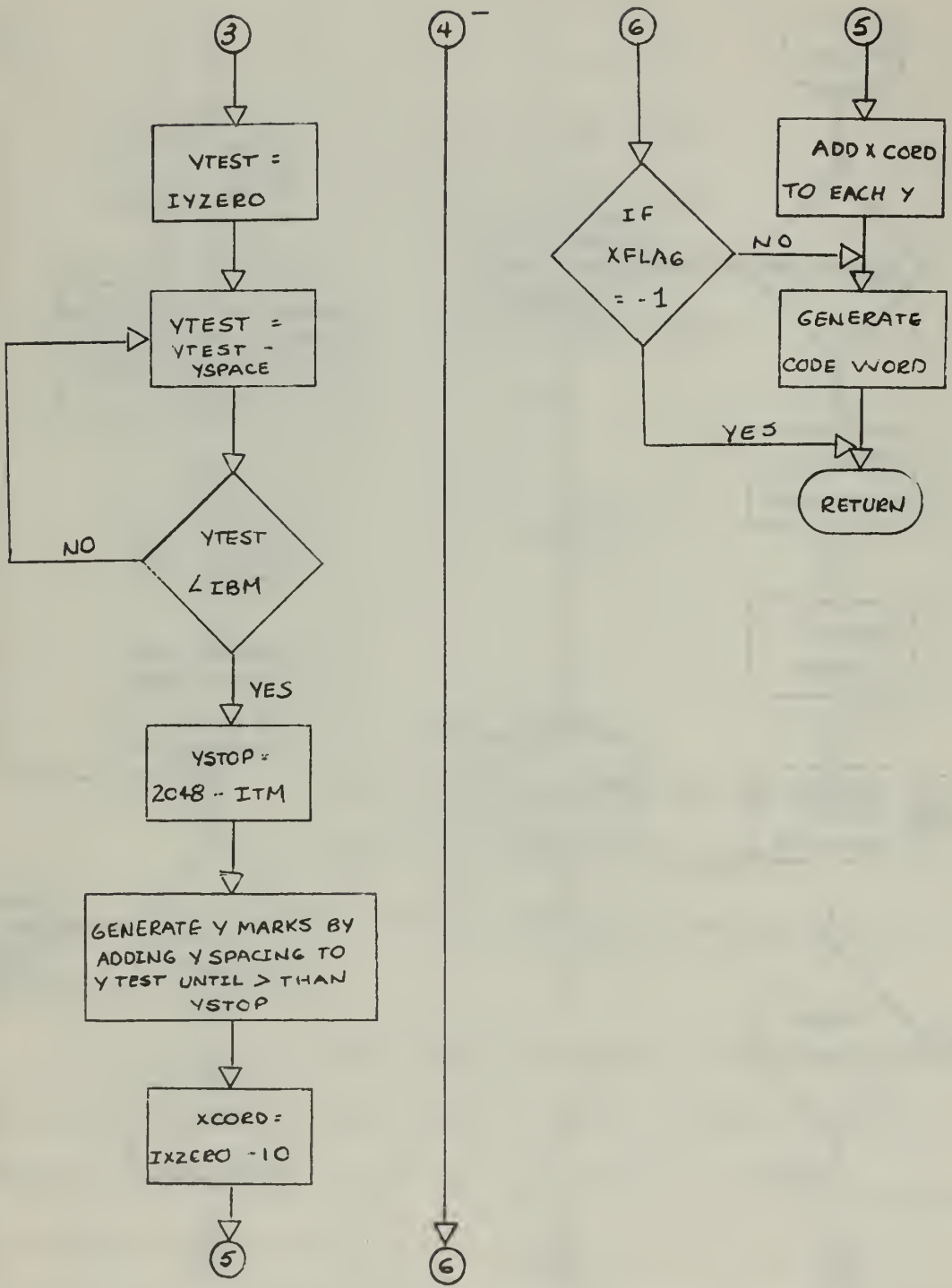
Discussion of flowchart:

Upon entry, the value of IXZERO is checked to see if it is on the display area. If it is, the spacing of the marks along the X axis is computed. This value is subtracted from IXZERO until the left margin is reached. Then the X spacing is added back in and the values are stored in the array ISTNG. Then the Y coordinate is added back to each value. The same procedure is followed to obtain the Y marks. The code word is then generated.



FLOWCHART 9

SUBROUTINE AXIS ARRAY



FLOWCHART 9 (CONT.)

SUBROUTINE AXIS ARRAY

10. ASSEMBLE - This subroutine assembles the data strings into a continuous loop to refresh the delay. This subroutine has a variable number of arguments. The subroutine is called in the following manner:

CALL ASMBLE (NCURV, IBREAK, ICMD, ICODE1, ICODE2...ICODEN)

Note: The dots would not actually be present, but indicate that there might be any number of arguments.

Description of parameters:

NCURV - The number of code words in the entering arguments.

IBREAK - A cell address where the Curve Add subroutine will insert a later curve calculated by this subroutine.

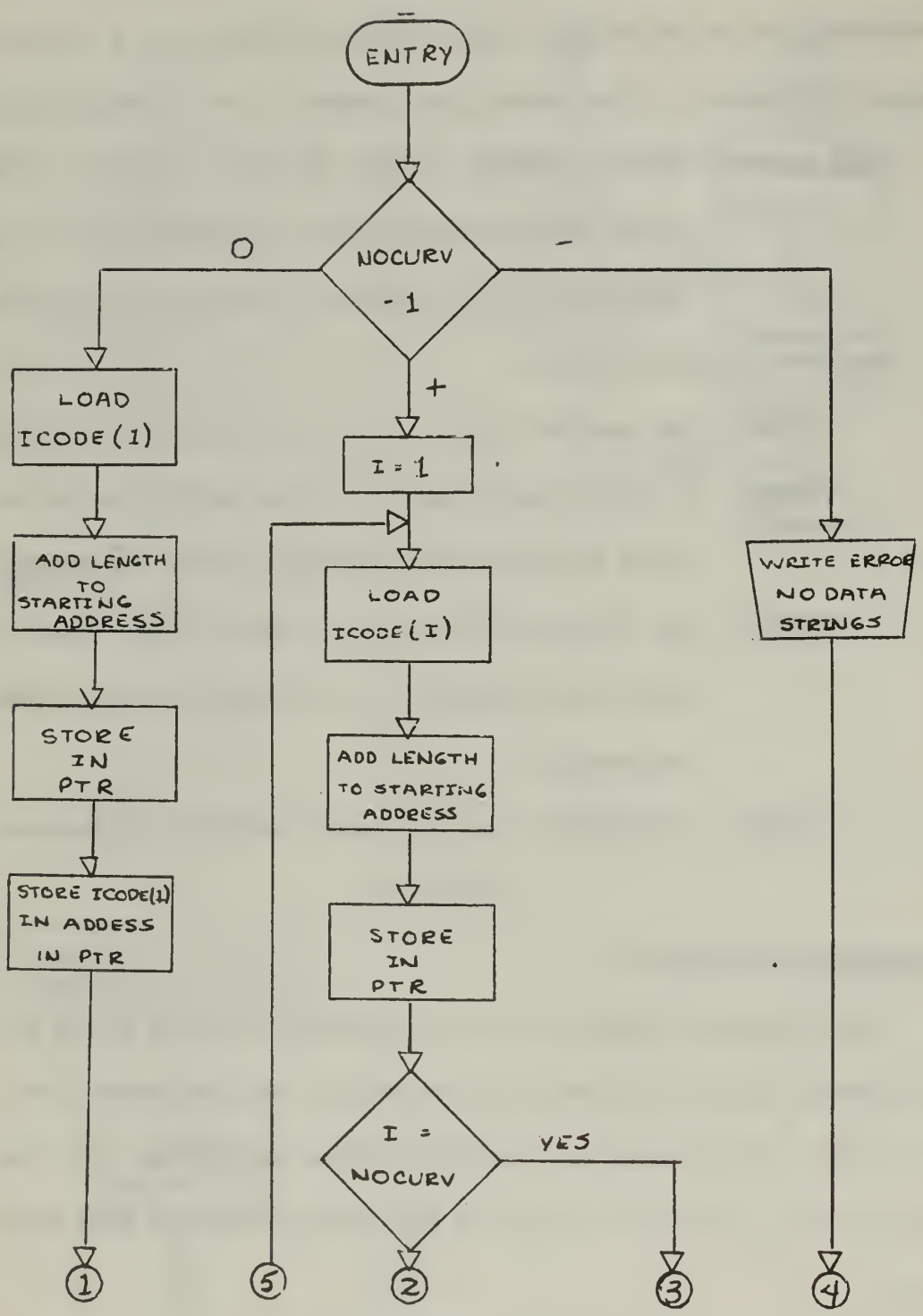
ICMD - The starting address of the refresh loop. This is the word transmitted to the display unit to initiate the display.

ICODE1...ICODEN - The code words generated by various subroutines.

Discussion of flowchart:

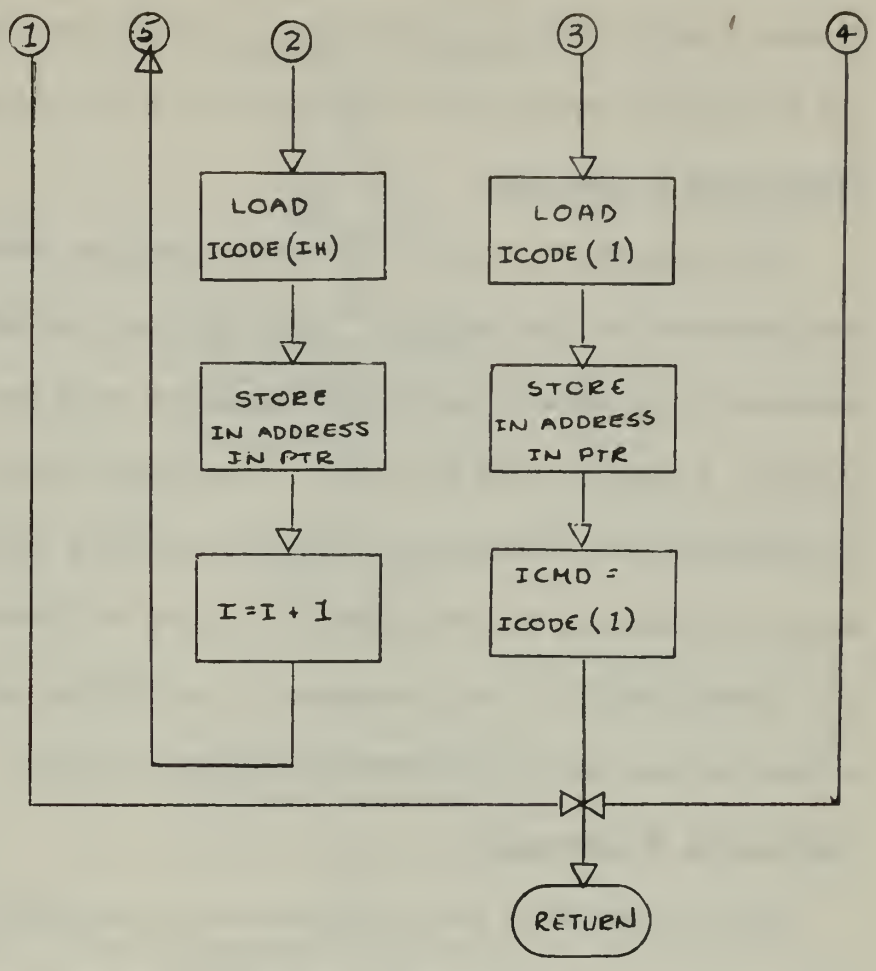
Each code word contains the starting address and the length of a data string. The second code word is added as the last word in the data string. This process is repeated for each data string. The last word in the last data string contains the code word for the first data string.

11. CURVE CHANGE - This element is not Fortran callable. It is entered by depressing the appropriate function switch on the display control. After the function switch is depressed, the light pen is used



FLOWCHART 10

SUBROUTINE ASSEMBLE



FLOWCHART 10 (CONT.)

SUBROUTINE ASSEMBLE

to designate the point to be changed. The light pen is then moved to the new location and the barrel switch is again depressed. The vertical and horizontal coordinates are changed to the new value.

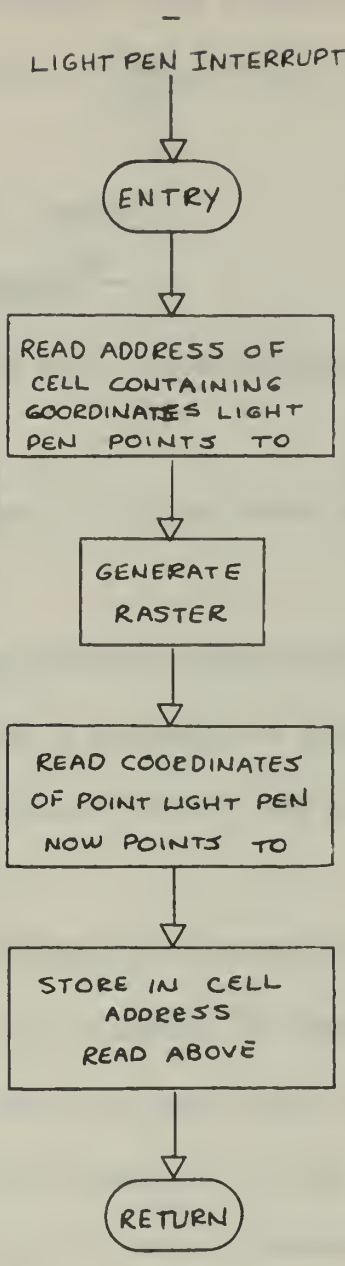
Description of flowchart:

The subroutine is entered through an interrupt location. The light pen interrupts are then enabled. When the light pen barrel switch is depressed, the address of the cell containing the X and Y coordinates is read. A raster is then generated. The second time the barrel switch is depressed, the coordinates of the new point are entered in the cell determined the first time the light pen switch was depressed.

12. CURVE DELETE - This subroutine is not Fortran callable. It is entered by depressing the appropriate function switch.

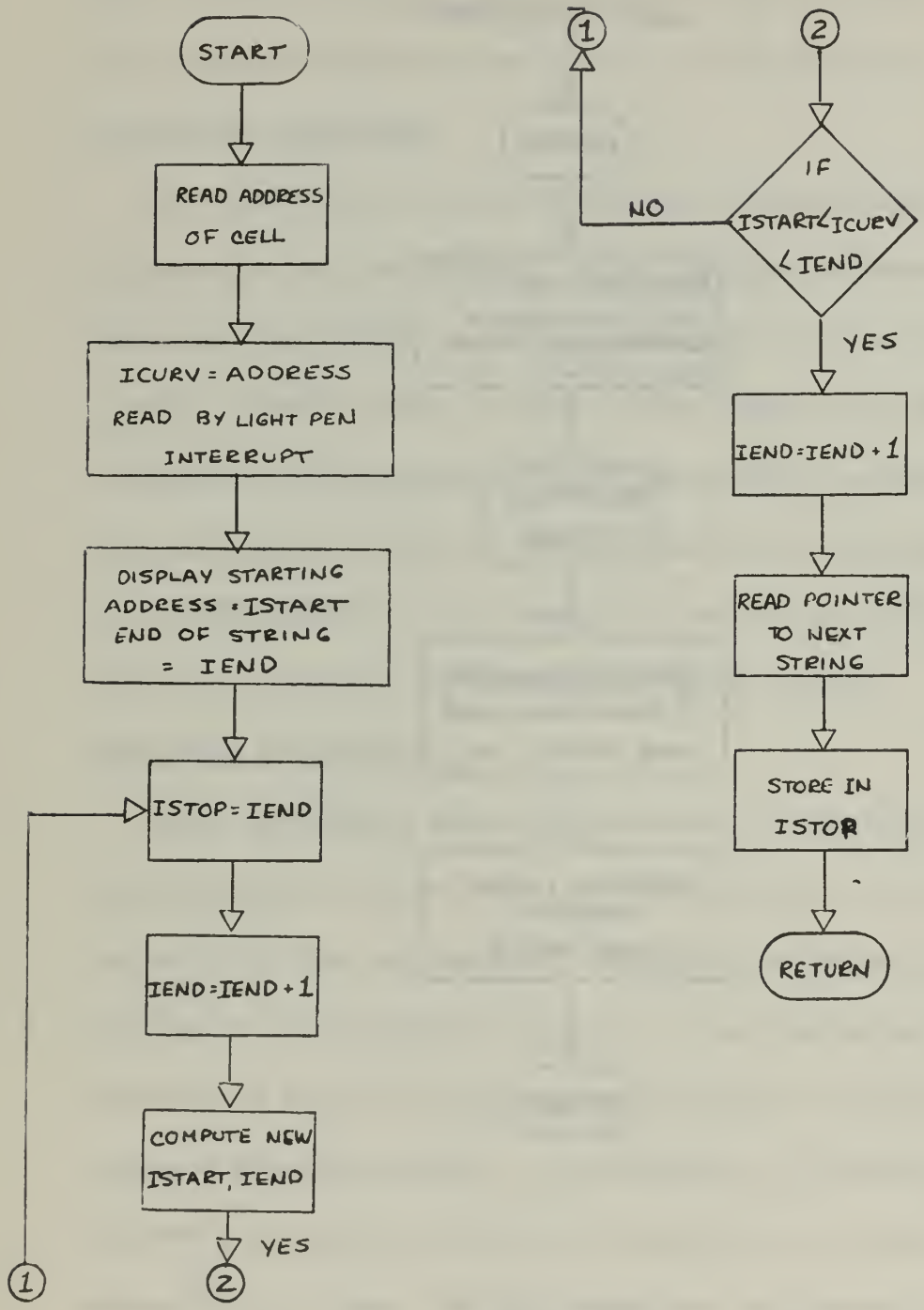
Discussion of flowchart:

When the function switch is depressed, the starting address of this subroutine is placed in cell 205 (the light pen interrupt cell). Upon arrival of the light pen interrupt, the address of the cell containing the coordinates of the light pen are read in. The starting and ending addresses of the first data string are found and the address read in is compared with these values. If the address is not within these limits, this was not the curve pointed to. The process is repeated until the proper string is found. String Pointers are then rearranged to skip the deleted data string in the refresh loop.



FLOWCHART 11

SUBROUTINE CURVE CHANGE



FLOWCHART 12

SUBROUTINE CURVE DELETE

13. CURVE ADD - This subroutine adds a data string at some time after the display has been initiated. The subroutine is called in the following manner:

CALL CURVAD (ICODE, IBREAK)

Description of parameters:

ICODE - The code word corresponding to the data string to be added.

IBREAK - The break in point generated by subroutine assemble.

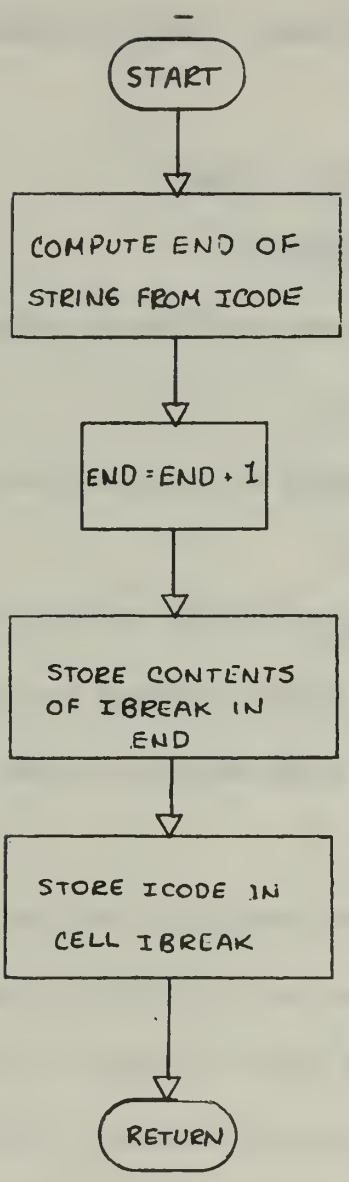
Discussion of flowchart:

The contents of the cell IBREAK contains the pointed connecting the two data strings. This word is removed and placed at the end of the new data string and the code word for the new data string is placed in the cell IBREAK.

14. CURVE DRAW - This subroutine uses the light pen to draw curves. It is not Fortran callable, but is entered by depressing the appropriate function switch. It will follow the light pen marking each coordinate that differs from the previous coordinate by fifty raster units. The subroutine is exited when a function switch corresponding to End Curve is depressed.

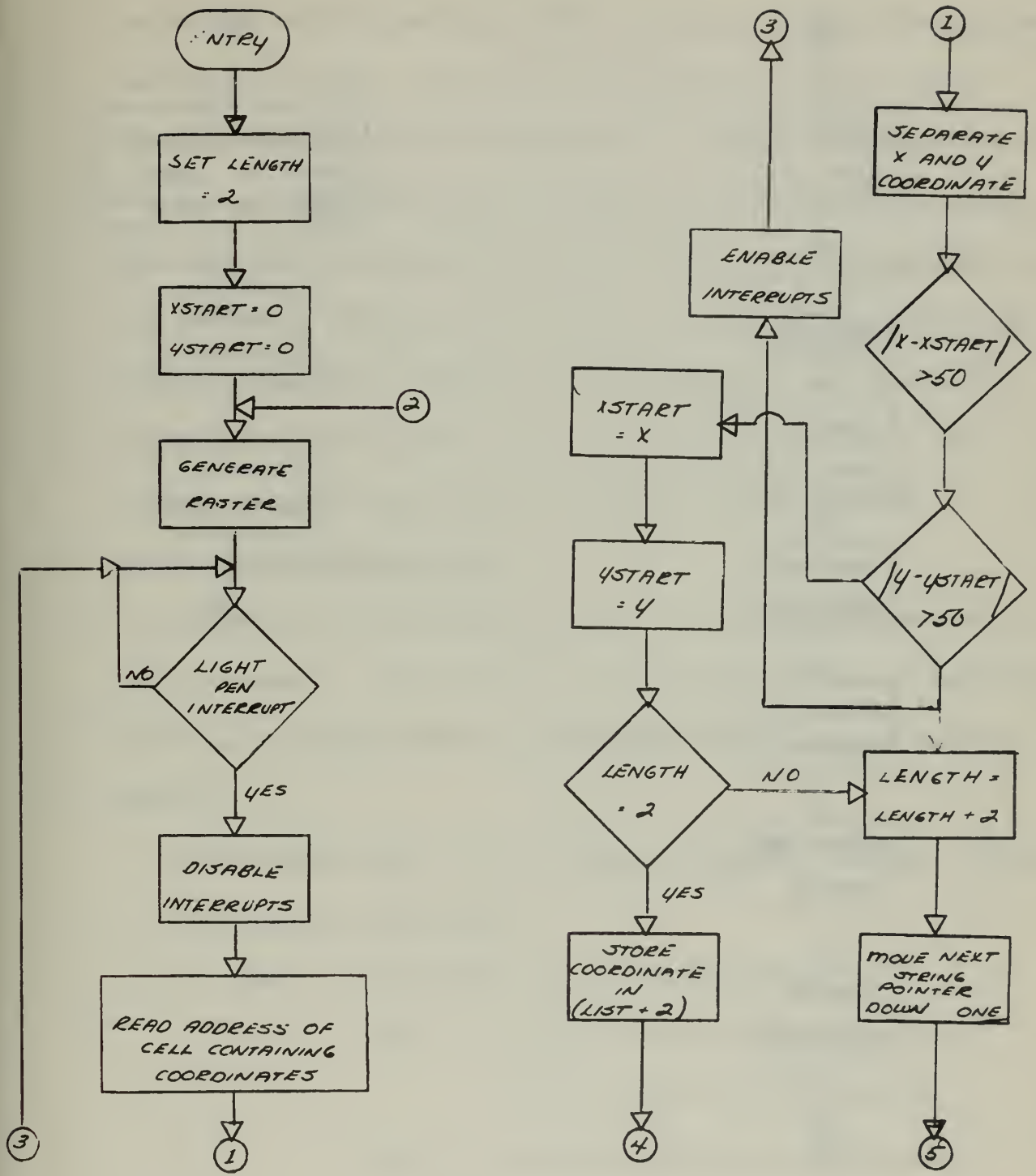
Discussion of flowgraph:

Upon entry, the data string length is set to equal two. The X and Y comparison coordinates (XSTART, YSTART) are initiated to zero. A raster is generated and the program waits for a light pen strike. When the light pen senses the raster, the address of the cell containing the X



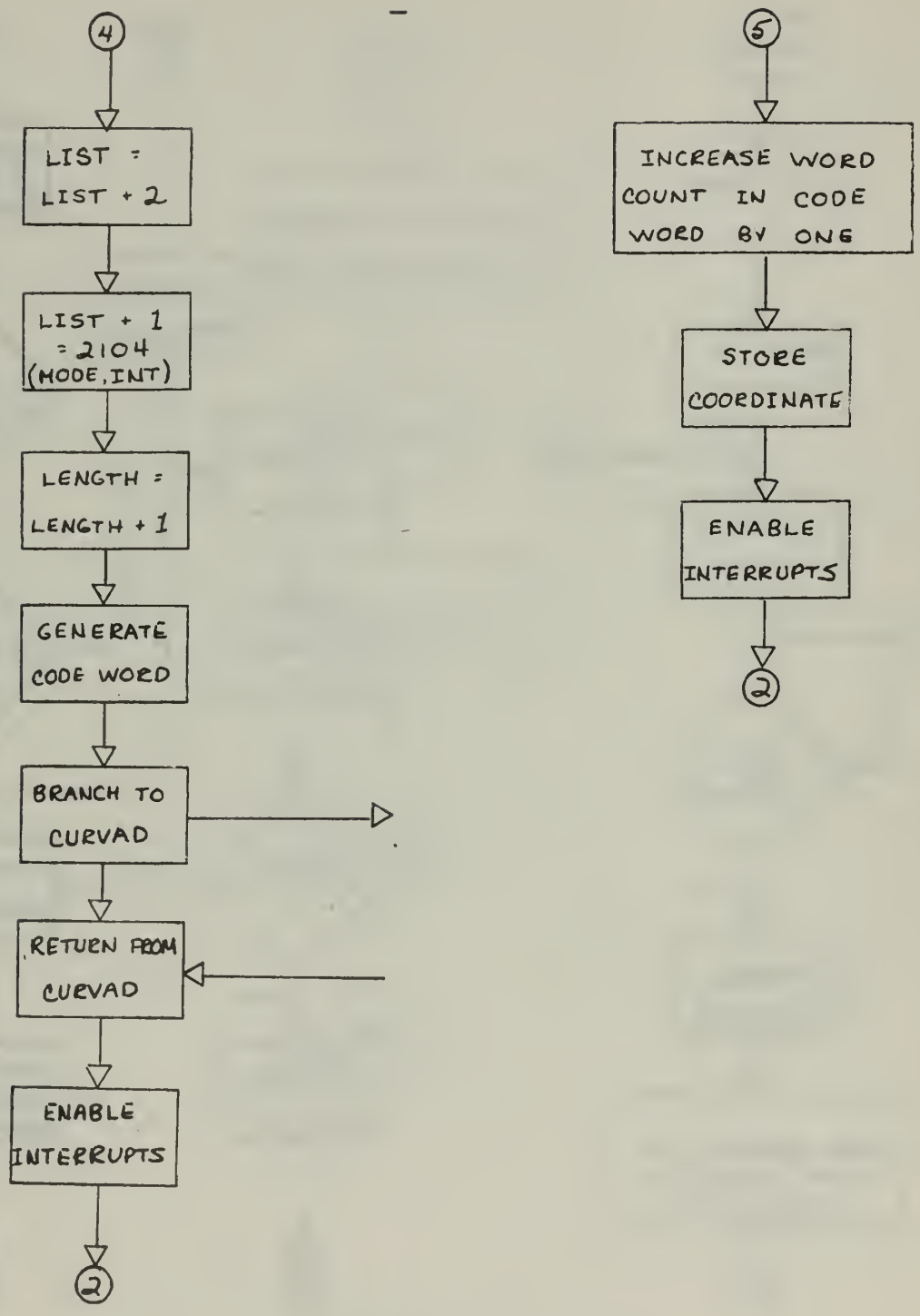
FLOWCHART 13

SUBROUTINE CURVE ADD



FLOWCHART 14

SUBROUTINE CURVE DRAW



FLOWCHART 14 (CONT.)

SUBROUTINE CURVE DRAW

and Y coordinates is read and the X and Y coordinates are found. These are checked with the previous X and Y coordinates to see if they change by fifty raster units. If not, the new coordinate is too close to the previous coordinate and the routine waits for another light pen strike to compare. If either the X and Y coordinate changes by fifty raster units, the coordinates are added to the list. If this is the first coordinate in the data string, the first two words are generated. The subroutine then updates its counters, generates the string code word and branches to CURVAD to include the data string in the display refresh loop. The program then goes back to wait for the next light pen interrupt.

Subset Tactical Situation Plot

1. ORIGIN FIX - This subroutine generates an origin from which to plot polar coordinates. This subroutine can be entered as often as necessary to change the current origin. The subroutine is called in the following manner:

CALL ORIGIN (ISET, ISCALE, ITIME, ICBR, SPDR, IXORGN, IYORGN)

Description of parameters:

ISET - Initializes the type of informations.

ISET = 0 ICRBR and SPDR are interpreted to be X and Y raster unit origin position.

ISET = 1 ICRBR and SPDR are interpreted to be course and speed of origin since the last plot.

ISET = 2 ICRBR and SPDR are interpreted to be polar direction and range origin has moved since last plot.

ISCALE - Integer number of nautical miles for entire display area.

ITIME - Integer number of minutes since last plot which is used if course and speed information is given.

ICRBR - Integer course or bearing (three digits - 000 to 359 degrees).

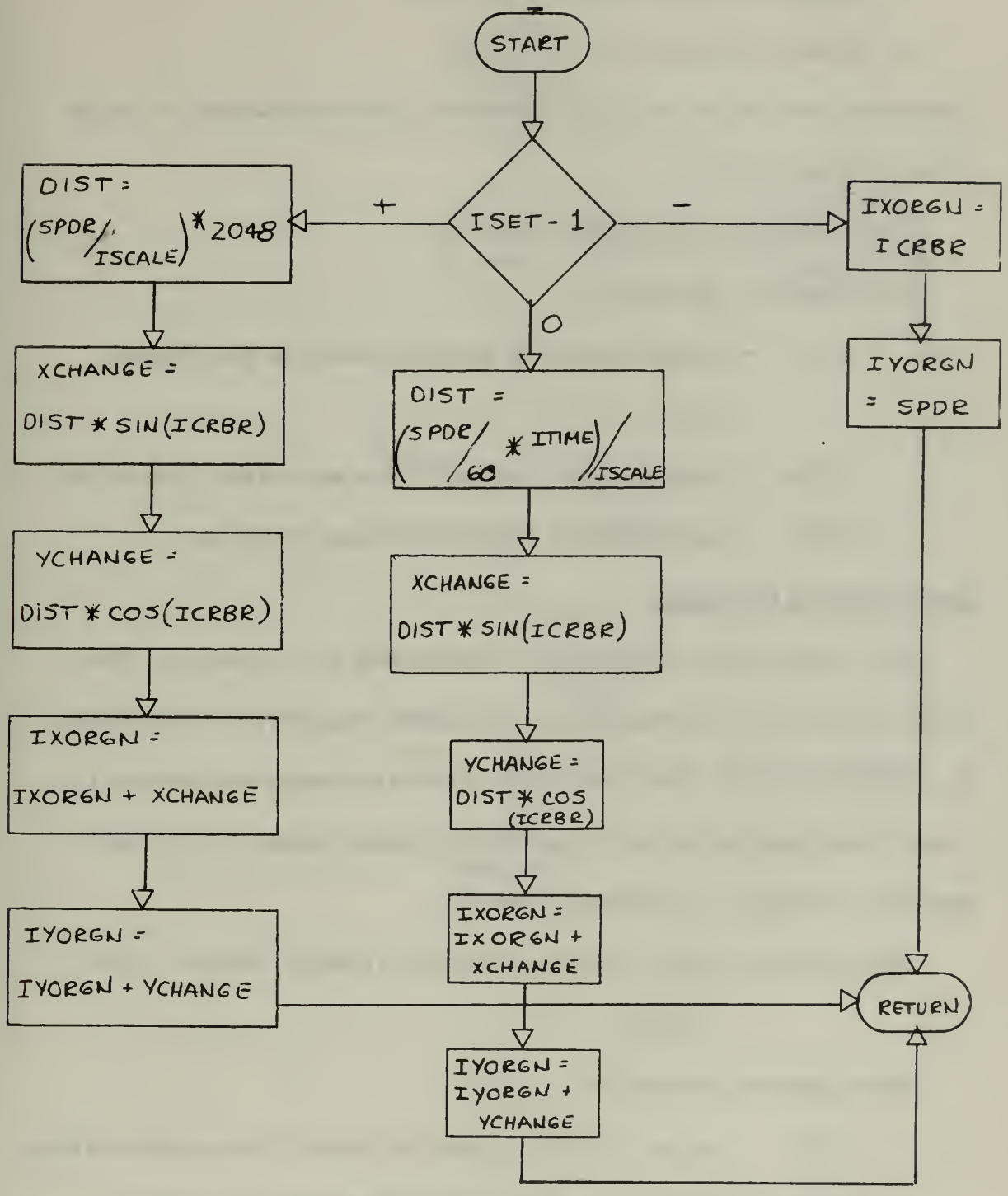
SPDR - Speed or distance in nautical miles.

IXORGN- Integer scaled X location of the present origin.

IYORGN- Integer scaled Y location of the present origin.

Discussion of flowchart:

Upon entry, ISET is checked to determine its value. The subroutine then branches to the proper area to perform the indicated operations. If ISET = 0, ICRBR and SPDR become the X and Y origin coordinates. If ISET = 1, the distance covered is calculated. This distance is broken down by the sine and cosine functions to become the changes to the X and Y origin coordinates. These values are added to the present values of IXORGN and IYORGN to become the new X and Y origin locations. If ISET = 2, SPDR becomes the distance the origin has moved. This distance is reduced to X and Y increments by the sine and cosine functions. The increments are then added to the present X and Y origin coordinates to generate the new values of IXORGN and IYORGN.



FLOWCHART 15

SUBROUTINE ORIGIN FIX

2. PROBLEM SCALE - This subroutine calculates two scale factors:

1. Raster units per nautical mile, and

2. Raster units per yard.

These are used by the various subroutines. This subroutine is called in the following manner:

CALL PROBSC (NM, SFNM, SFYD)

Description of parameters:

NM - Integer number of nautical miles for full display area.

SFNM - Scale factor - nautical miles per display raster unit.

SFYD - Scale factor - yards per display raster unit.

Description of flowchart:

The argument NM is checked to insure that it is positive. The scale factors then become SFNM = NM/2048, and SFYD = SFNM/2000.

3. POSITION PLOT - This subroutine converts a range and bearing to X and Y coordinates and adds them to the present origin. This subroutine is called in the following manner:

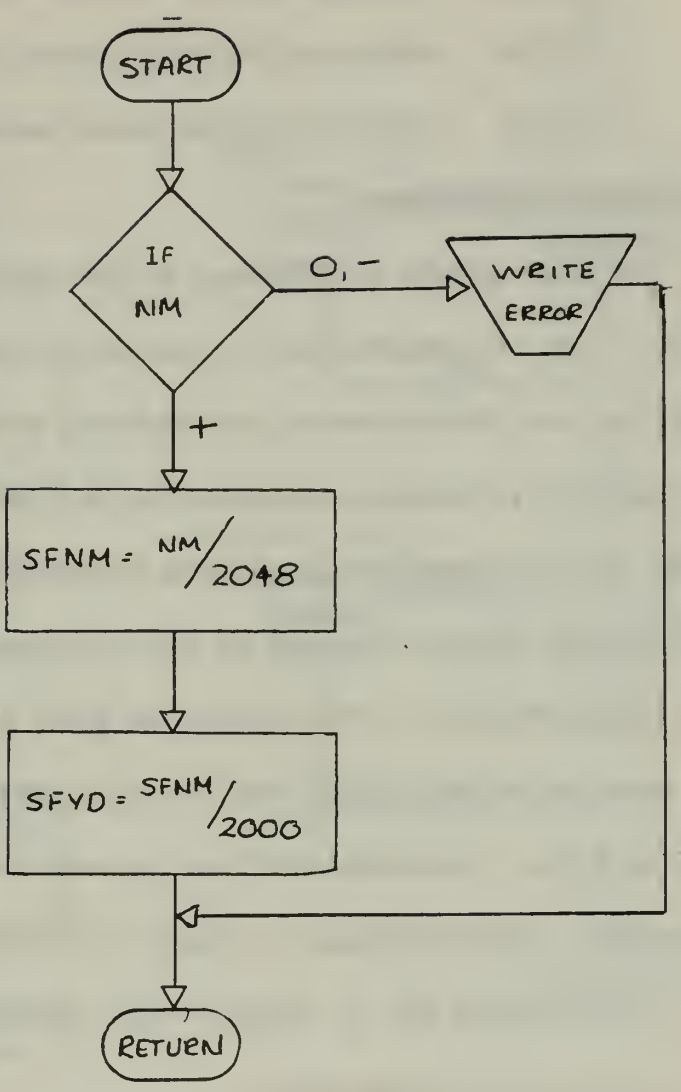
CALL POSPLOT (IRNG, IBRNG, IXORGN, IYORGN, SCALE, IXPOS, IYPOS)

Description of parameters:

IRNG - Integer number of yards of contact from present origin.

IBRNG - Bearing of contact from present origin. Integer number of degrees from 000 to 359.

IXORNG- Present horizontal location of origin.



FLOWCHART #16

SUBROUTINE PROBLEM SCALE

IYORNG- Present vertical location of origin.

SCALE - Multiplication factor to convert range to raster units.

IXPOS - Horizontal location (raster units).

IYPOS - Vertical location (raster units).

Discussion of flowchart:

The range (IRNG) is multiplied by the scale factor (SCALE). The scaled range is multiplied by the sine of the bearing for the relative Y coordinate and the cosine for the relative X coordinate. The values are truncated to integers and added to the X and Y origin values. If either the X or Y position is negative or greater than 2048 the location is not on the display area and an error message will be typed out.

4. CHARACTER PLOT - This subroutine plots any ASC II character at any location on the display area. Special characters and figures can also be drawn. See Appendix D for the method of generating special characters. This subroutine is called in the following manner:

CALL CHPLOT (IX, IY, ISIZE, ICHR, LENGTH, INT, ISTNG, ICODE)

Description of parameters:

IX - Integer X coordinate at left edge of character.

IY - Integer Y coordinate at bottom edge of character.

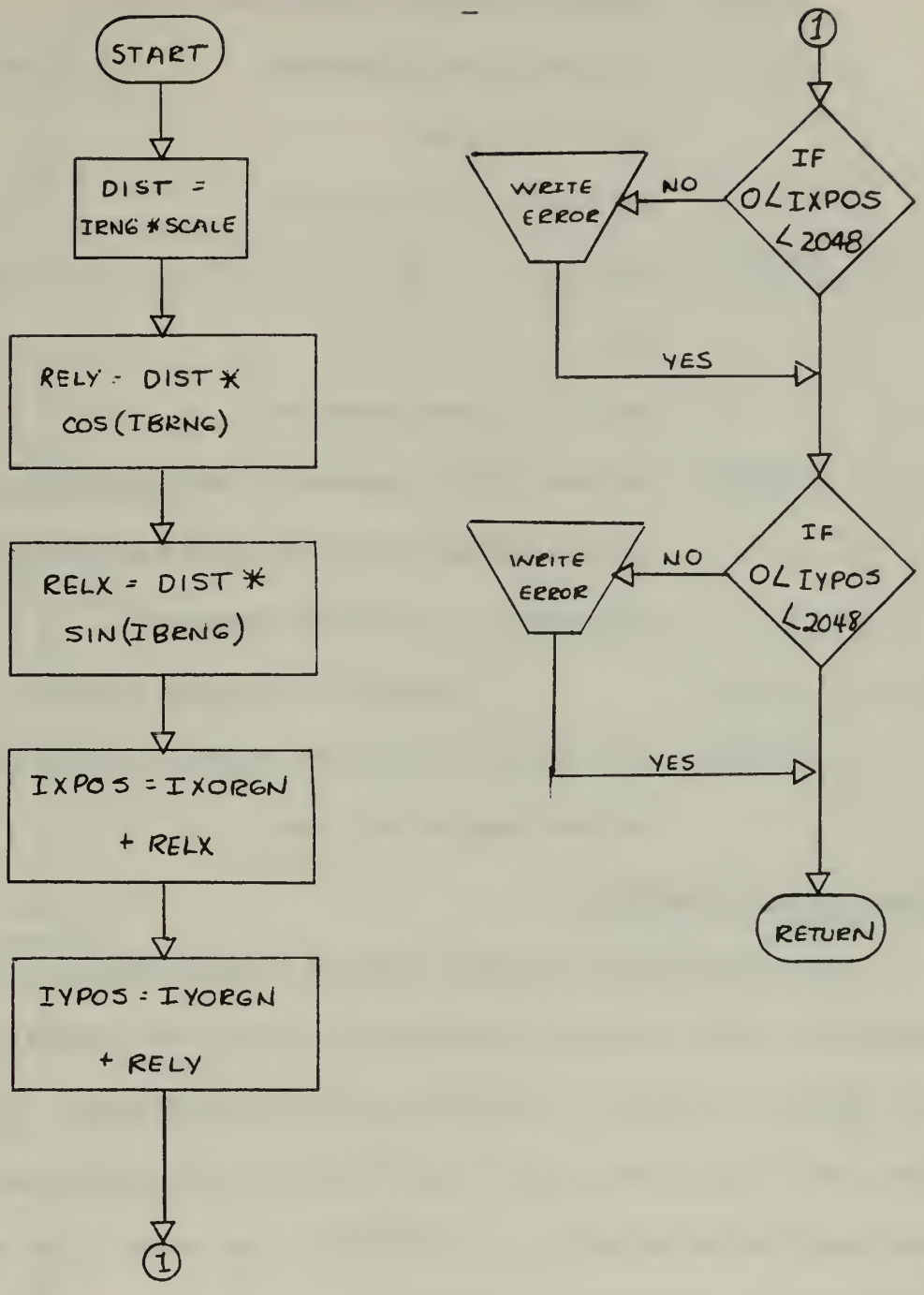
ISIZE - Size of the characters (if standard).

ISIZE = 0 small size

ISIZE = 1 medium size

ISIZE = 2 large size

ISIZE = 3 special character



FLOWCHART #19

SUBROUTINE POSITION PLOT

ICHR - Single digit number to specify the character.

LENGTH- Length of special character data string.

INT - Intensity of the characters.

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

INT = 08 most intensity

ISTNG - The data string generated by this subroutine. It must be dimensioned in the user's program.

Dimension: 4 if standard character,

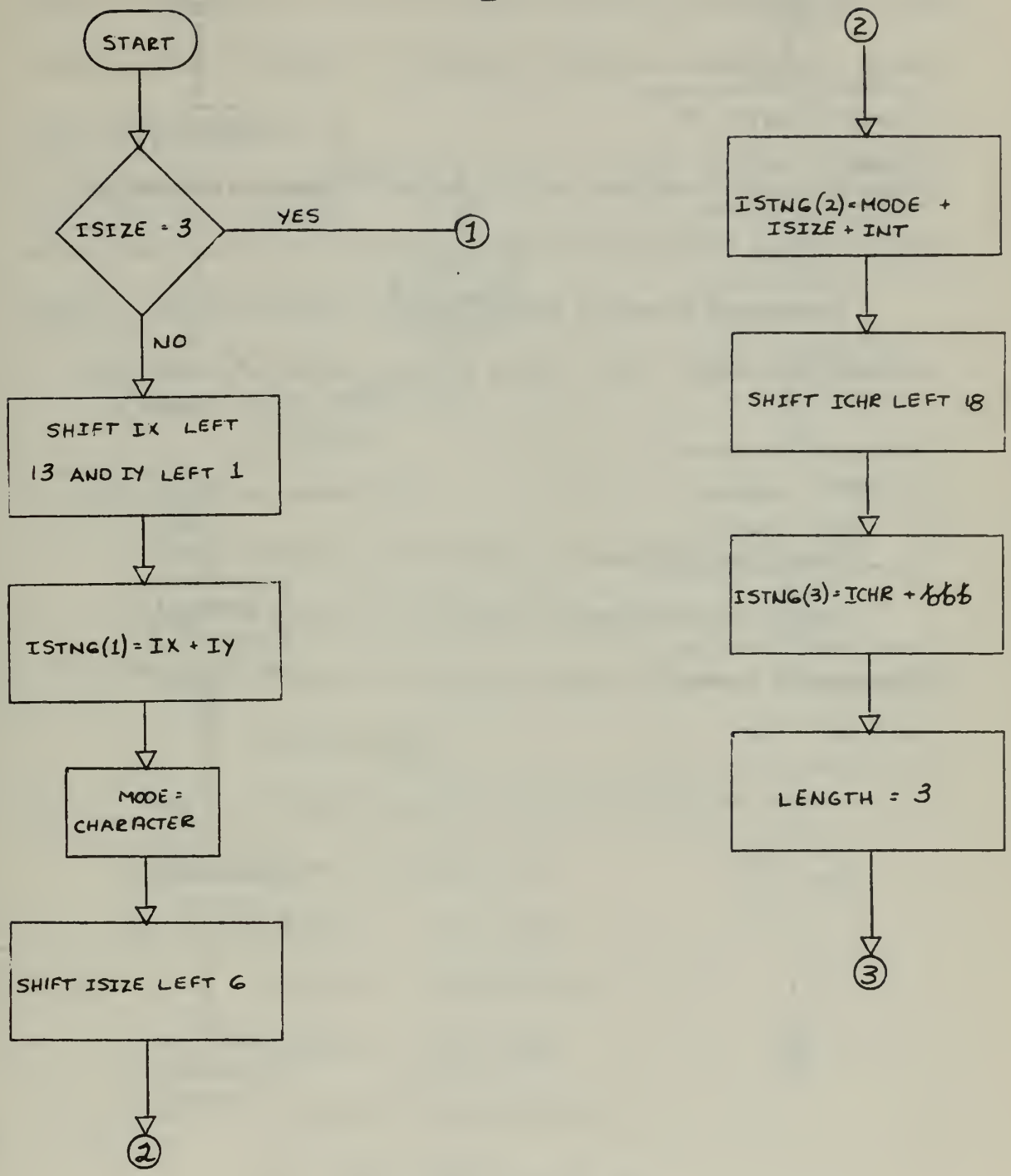
LENGTH +3 if special character

ICODE - Code word specifying the start of the data string and the length of the string.

Discussion of flowchart:

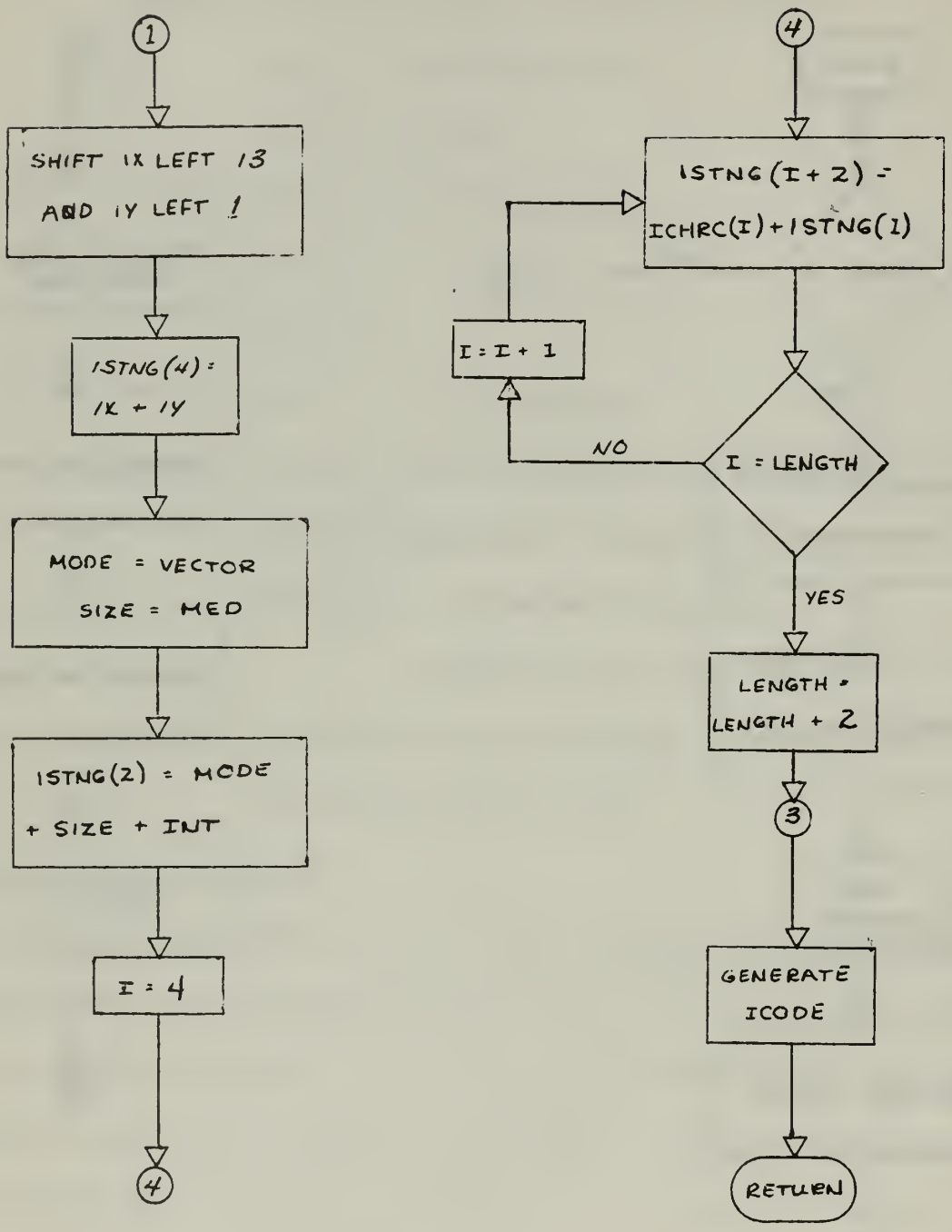
Upon entry, ISIZE is checked to see if it equals three. If it does equal three, the subroutine recognizes it as a special character. If it is a standard character, IX and IY are packed into ISTNG(1). The mode, size, and intensity are packed into ISTNG(2). The desired character and three spaces are packed into ISTNG(3). The length of the data string is three. ICODE is generated.

If a special character is to be generated, ICHR is taken to be an array (as outlined in Appendix D). LENGTH is the number of computer words in the array. IX and IY are packed and placed in ISTNG(1). The



FLOWCHART #18

SUBROUTINE CHARACTER PLOT



FLOWCHART 18 (CONT.)

SUBROUTINE CHARACTER PLOT

mode is set to vector, and ISTNG(2) is generated. ISTNG(1) is then added to each value in the array ICHR and they are stored in the corresponding WORD in ISTNG. The length of ISTNG is LENGTH +2. ICODE is then generated.

5. INFORMATION PLOT - This subroutine plots the course and speed of a contact one-fourth inch to the right of the present position of the contact. This subroutine is called in the following manner:

CALL INFPLT (IPRESX, IPRESY, ICRSE, ISPD, ISIZE, INT, ISTNG, ICODE)

Description of parameters:

IPRESX - Present X coordinate of target(Integer value).

IPRESY - Present Y coordinate of target(Integer value).

ICRSE - Course of contact in degrees (Integer value between 000 and 359).

ISPD - Speed of contact in knots (Integer value).

ISIZE - Size of the characters.

ISIZE = 0 small size

ISIZE = 1 medium size

ISIZE = 2 large size

INT - Intensity of the characters.

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

INT = 08 most intensity

ISTNG - Data string containing above information for display.

ICODE - Code word containing length of data string and the starting address.

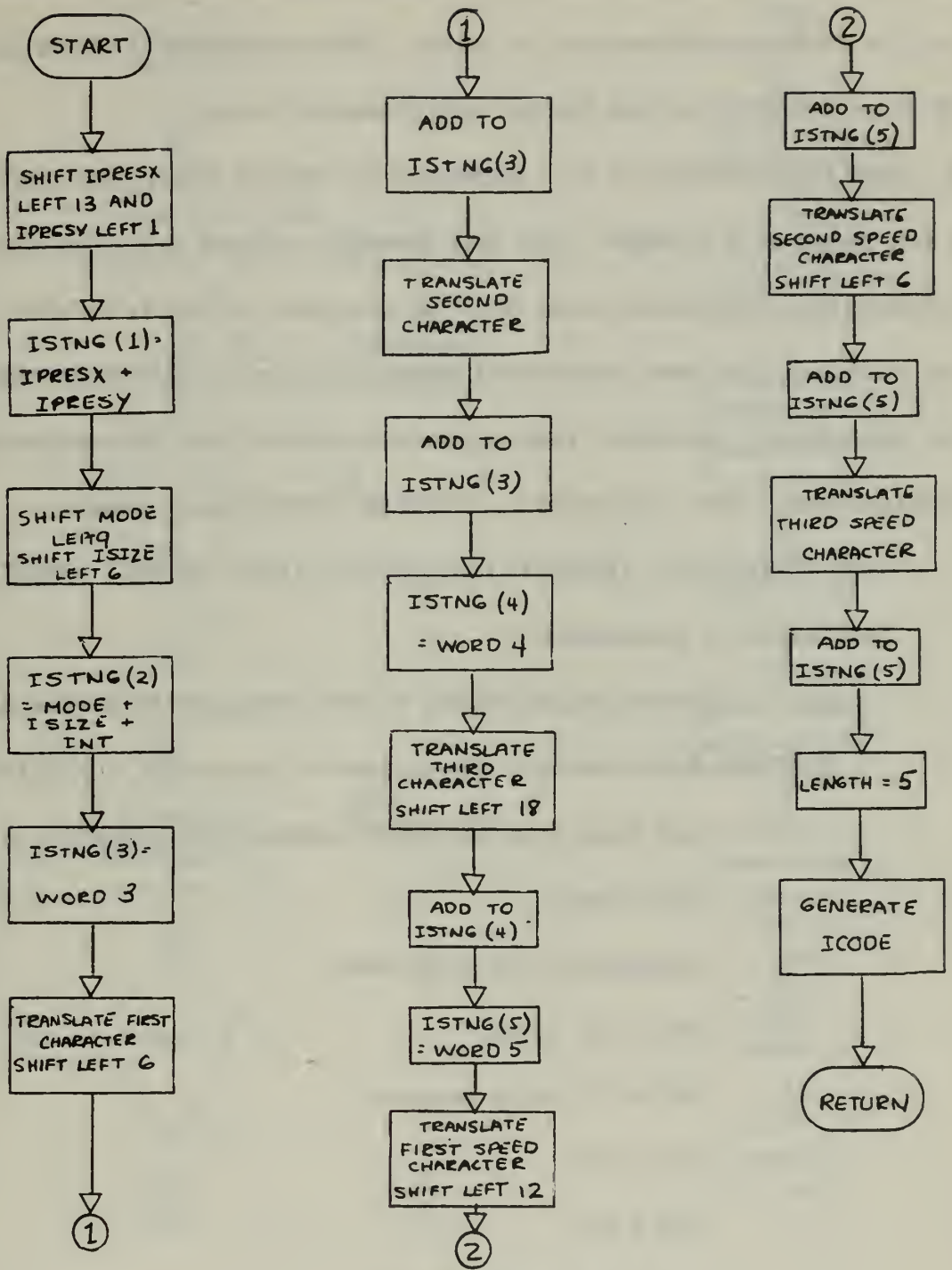
Discussion of flowchart:

The basic string generated will have five words. The following string exists in the subroutine and is used to generate the data string ISTNG:

WORD	CONTENTS	REMARKS
WORD 1	BLANK	IPRESX and IPRESY are packed in this word.
WORD 2	MODE | ISIZE | INT	MODE is set to the character ISIZE and INT are packed into the appropriate bits.
WORD 3	C | = | C_1 | C_2	C_1, C_2, and C_3 represent the three digit course.
WORD 4	C_3 | b | b | S	
WORD 5	= | S_1 | S_2 | S_3	S_1, S_2, and S_3 are the speed digits.

TABLE 2

DATA STRING ISTNG



FLOWCHART 19

SUBROUTINE INFORMATION PLOT

The present values of X and Y (IPRESX and IPRESY) are packed into the first word of ISTNG. The second and following words are copied and the necessary characters are added. The characters of ICRSE and ISPD are converted to the display unit character codes.

6. PAST INFORMATION - This subroutine is entered every time there is a new position of a target. The data string is updated to reflect new information. Information older than the specified amount is deleted. If the track has just been started and there is less past information than the track length specifies, then only the available past information will be displayed. This subroutine is called in the following manner:

CALL PSTINF (ITL, IENTNO, INT, NEWX, NEWY, ISTNG, ICODE)

Description of parameters:

ITL - Track length number of past positions to be plotted.

IENTNO- Entry number. The number of times this subroutine has been entered for this particular information or code word.

INT - Intensity of the characters.

INT = 00 blank

INT = 01 least intensity

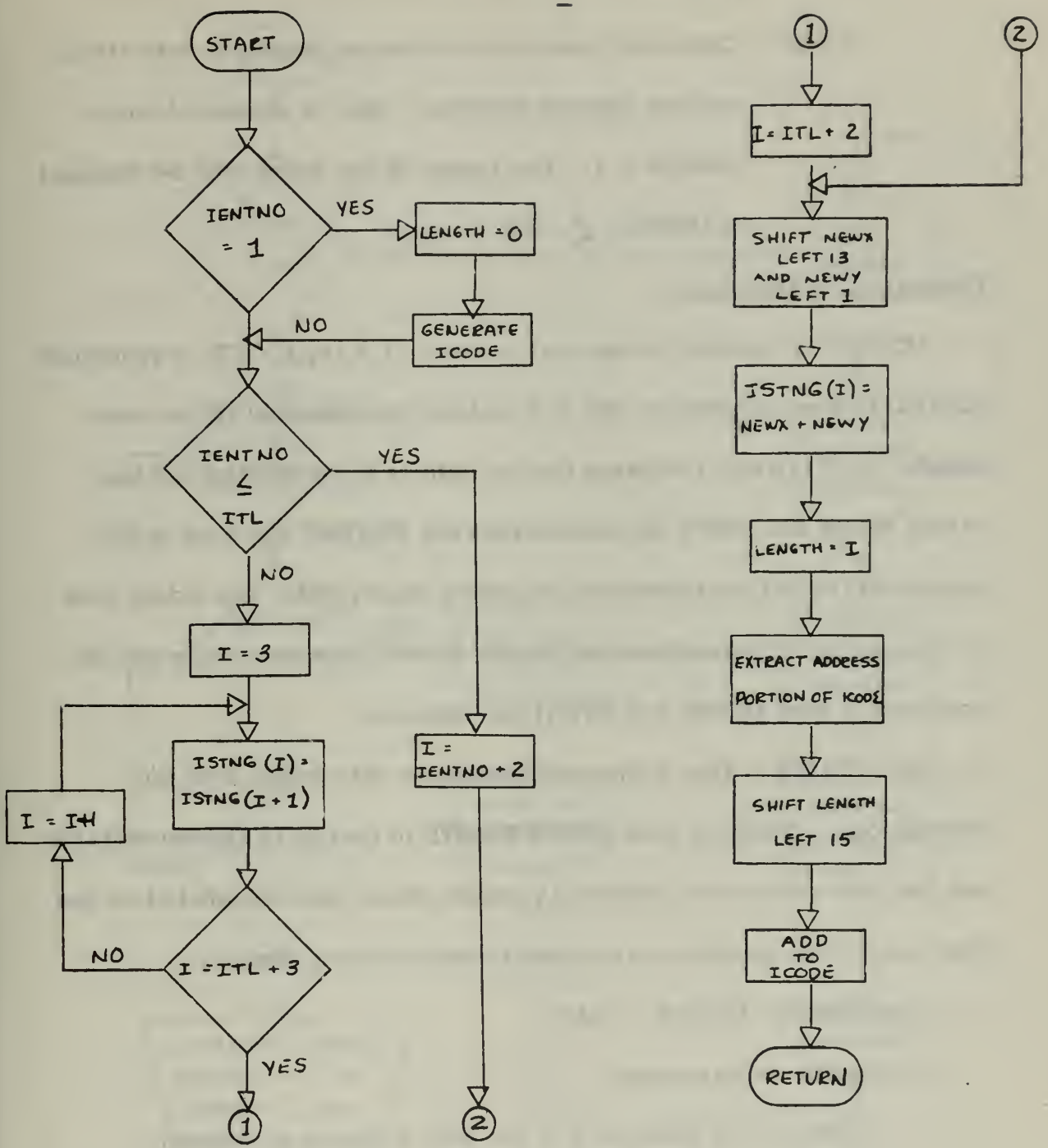
INT = 02 . .

INT = 04 . .

INT = 08 most intensity

NEWX - New X coordinate to be added to data string.

NEWY - New Y coordinate to be added to data string.



FLOWCHART 20

SUBROUTINE PAST INFORMATION

ISTNG - Data string of length ITL +2 which contains past information to be plotted.

ICODE - Code word generated indicating length of data string and the starting address. This is generated only if IENTGN = 1. The length of the string will be changed if IENTNG < ITL.

Discussion of flowchart:

IENTNO is checked to see if it is one. If it is, ICODE is generated. IENTNO is then checked to see if it is less than or equal to the track length. If it is, this indicates that no data is to be deleted and the values NEWX and NEWY are packed into the (IENTNO +2) word of the string. If IENTNO is larger than the track length (ITL), the oldest data is removed, all intermediate values are moved up on the string and the most recent data (NEWX and NEWY) are inserted.

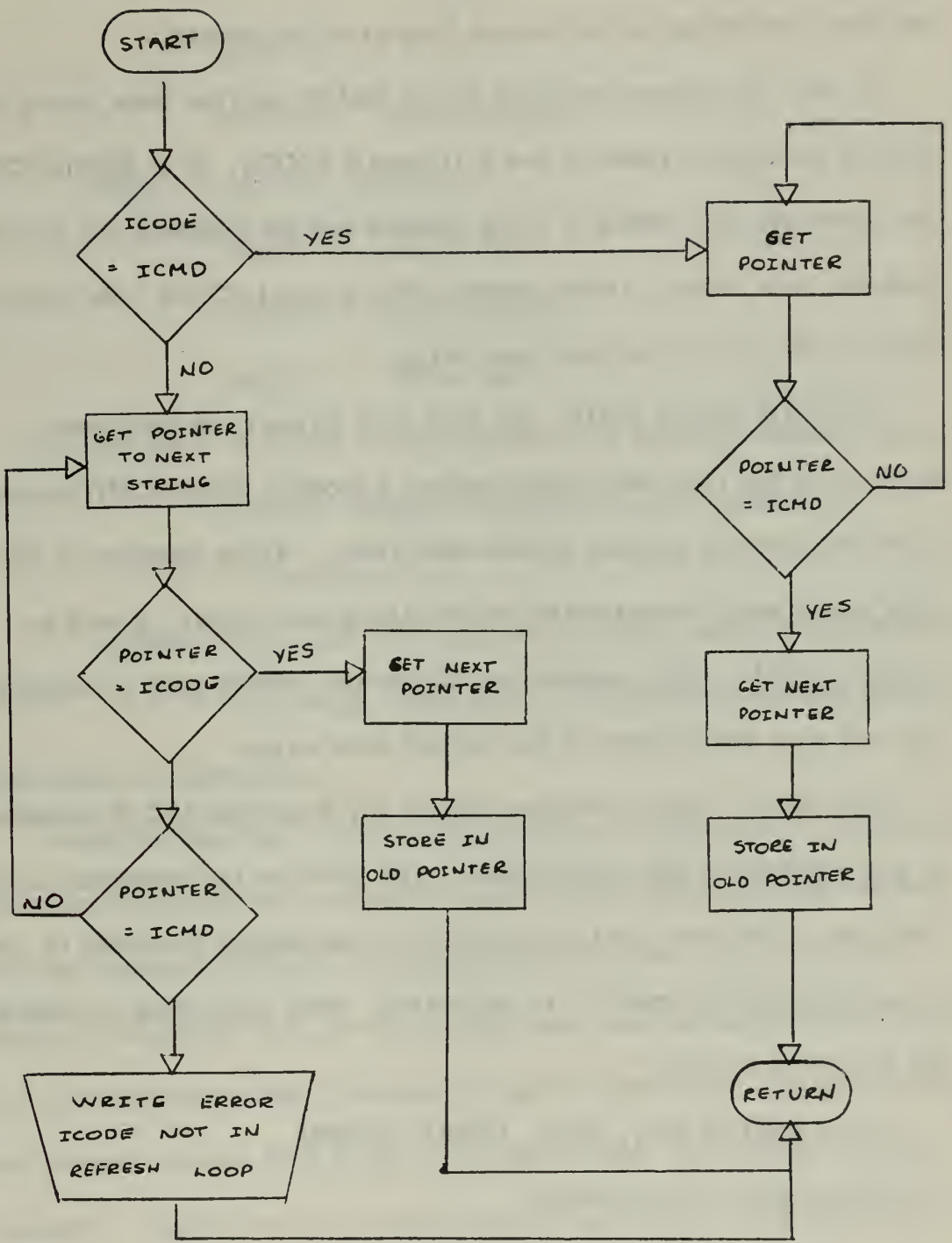
7. UNIT DELETE - This subroutine removes a data string from the refresh loop. It differs from CURVE DELETE in that is is Fortran callable and the data string to be deleted is named rather than designated by the light pen. This subroutine is called in the following manner:

CALL UNITDL (ICODE, ICMD)

Description of parameters:

ICODE - The code word of the data string to be deleted.

ICMD - The command word generated by subroutine ASSEMBLE.



FLOWCHART 21

SUBROUTINE UNIT DELETE

Discussion of flowchart:

Upon entry, ICODE is checked to see if it equals ICMD indicating the first data string in the refresh loop is to be deleted.

If not, the subroutine looks at the end of the first data string and checks the pointer there to see if it equals ICODE. If it equals ICODE, the following data string is to be deleted and the pointers are moved to skip the data string. If the pointer did not equal ICODE, the subroutine looks at the end of the next data string.

If ICODE equals ICMD, the first data string is to be deleted. Removal of the first data string causes a problem since ICMD is made up of the starting address of this data string. Since deletion of this data string would necessitate reinitializing the display as well as changing ICMD, this subroutine changes the refresh loop by making the last data string point to the second data string.

8. UNIT ADD - This subroutine places any available ASC II character at any location on the display area. The location is designated by the light pen. After the point is designated, the display keyboard is used to indicate which symbol is to be plotted. This subroutine is called in the following manner:

CALL UNITAD (INT, ISIZE, ISTNG, ICODE)

Discussion of parameters:

INT - Intensity of characters.

INT = 00 blank

INT = 01 least intensity

INT = 02 . .

INT = 04 . .

INT = 08 most intensity

ISIZE - Size of characters.

ISIZE = 0 small size

ISIZE = 1 medium size

ISIZE = 2 large size

ISTNG - The data string generated by this subroutine which must be dimentioned in the user's program.

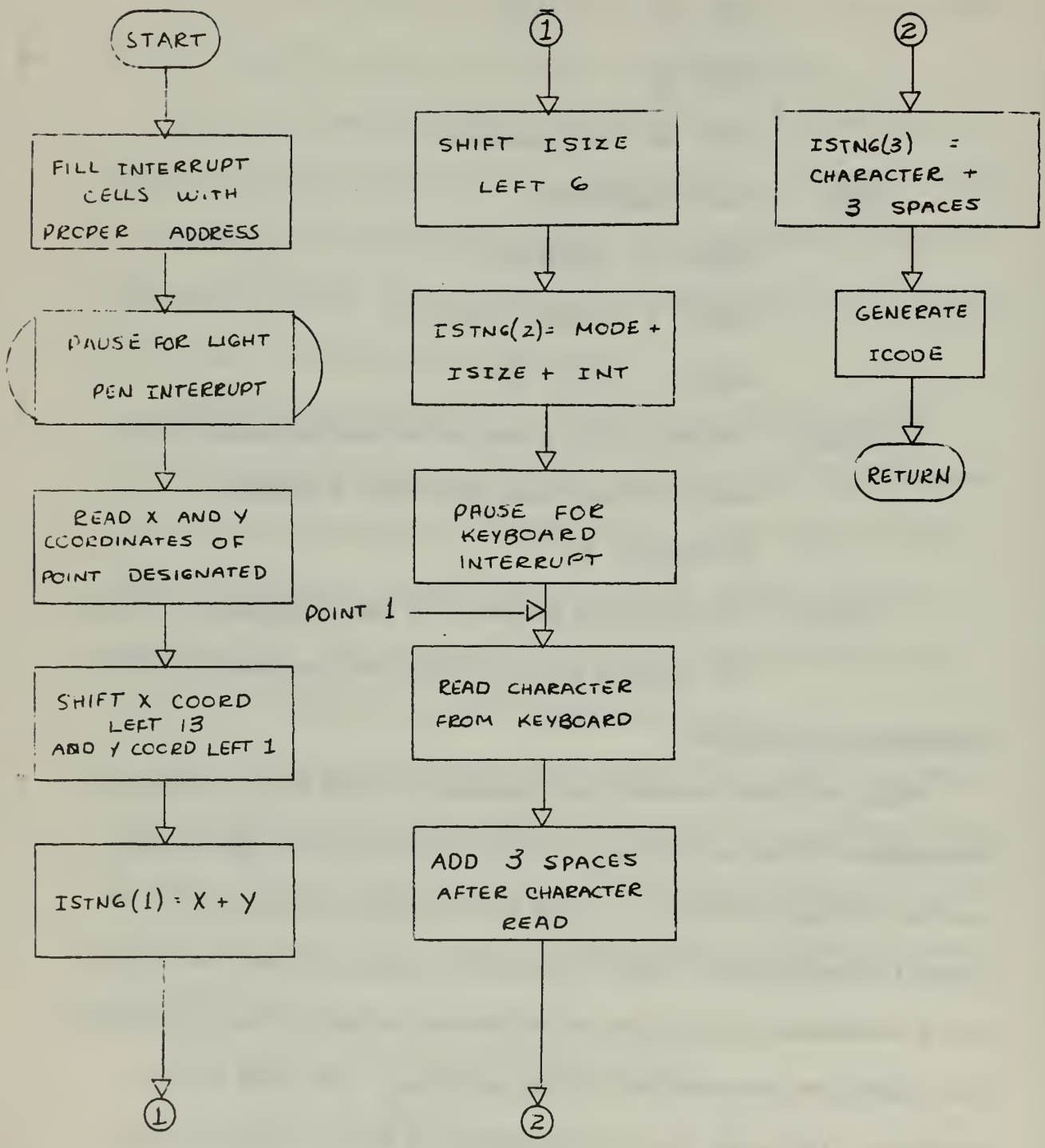
Dimension: 4.

ICODE - The code word generated by this subroutine containing the length of the data string and the starting address.

Discussion of flowchart:

When the function switch corresponding to UNIT ADD is depressed, the address of this subroutine is placed in the light pen interrupt cell and the address of Point 1 is placed in the keyboard interrupt cell. A raster is generated and, when the light pen interrupt is received, the X and Y coordinates of the location designated are read. These coordinates are packed into the first word of the data string. The mode is set to character. The mode, size, and intensity are packed into the second word of the data string. The character is read from the display keyboard and placed in the third word of the data string. ICODE is then generated.

9. ASSEMBLE - This subroutine is the same as subroutine ASSEMBLE discussed before.



FLOWCHART 22

SUBROUTINE UNIT ADD

The light pen interrupt occurs only when the light pen senses light on the display area. When drawing curves a raster or pattern of light must be generated on the display viewing area. There are several possible methods of accomplishing this, each of which has advantages in certain applications. One method would be to fill the entire viewing area with points. This is very rapid, but the core requirements are excessive. Another method would be to generate a line of points and move it down the display area. The third method uses an expanding square. This method is useful if the point to be designated is close to the origin of the expanding square. Drawing curves is an example of this since the origin of the expanding square is the last point designated.

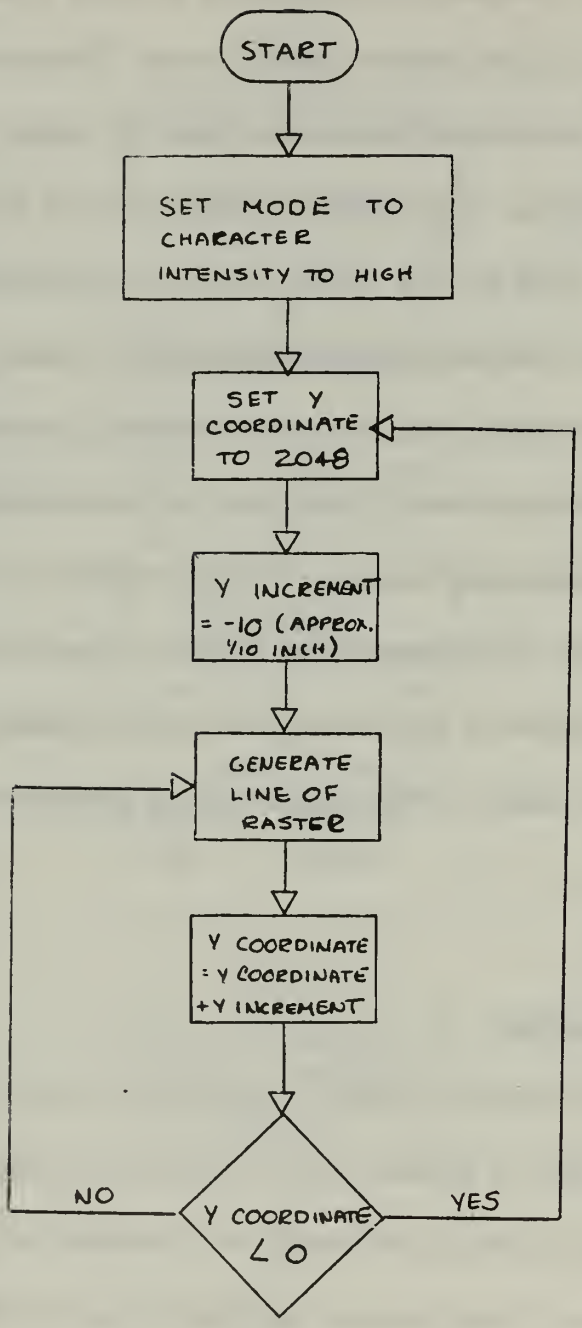
The second method is implemented due to the ease of programming and more versatile nature. This subroutine is called in the following manner:

CALL RASTER

Discussion of flowchart:

The subroutine sets the initial Y coordinate to 2048 (top of the display area). A line of points in the X direction is generated. A negative increment is added to the Y coordinate and another horizontal line of points is generated. This process is repeated until the Y coordinate is negative, indicating it is off the display area. The entire process is then repeated until a light pen interrupt is received.

The subroutines above provide for setting up the refresh loop and modifying it during the display process. The individual displays are initiated by the following Fortran callable subroutine:



FLOWCHART 23

SUBROUTINE RASTER

CALL DISPLAY (NO, ICMD)

Description of parameters:

NO - Number of display to be used.

NO = 1 display number 1 to be used.

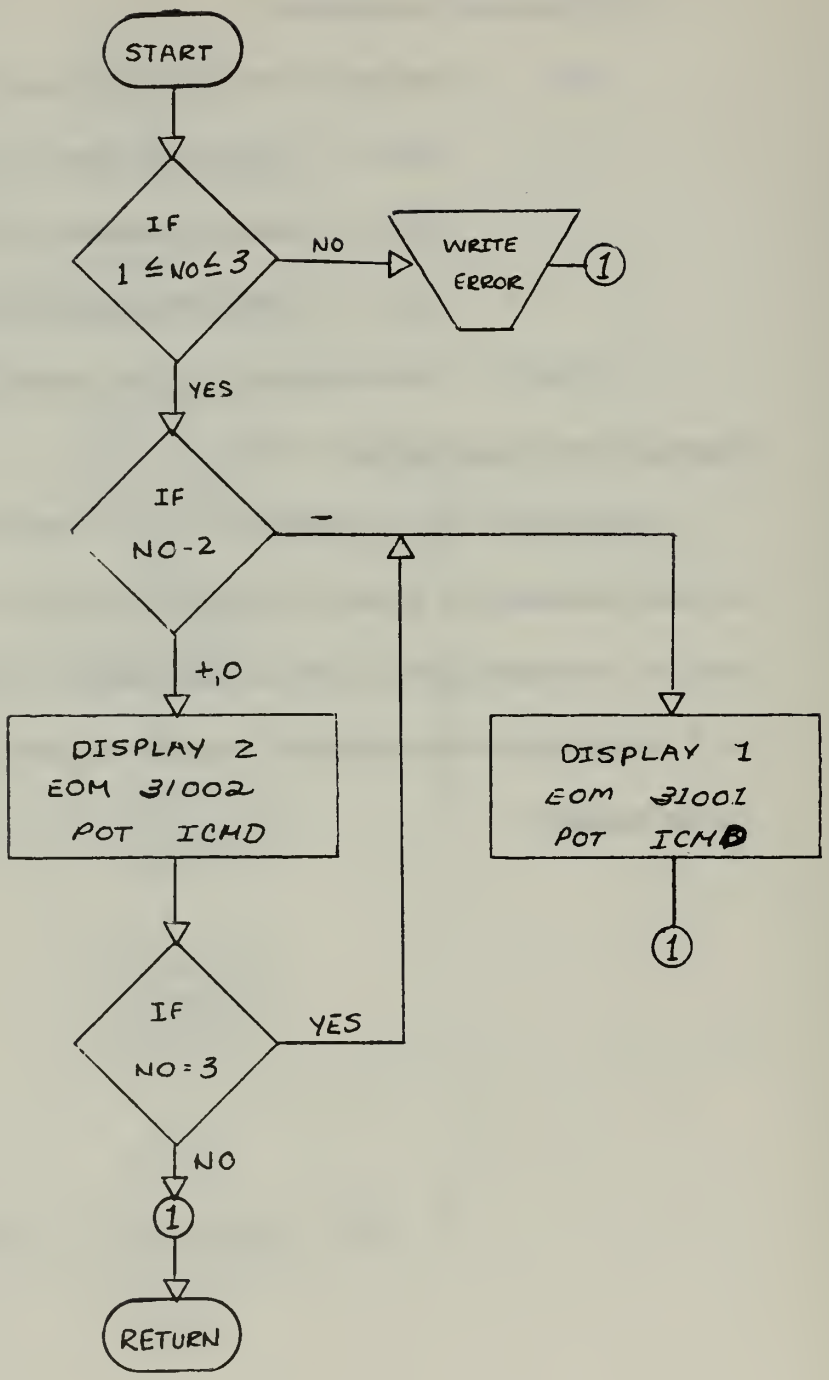
NO = 2 display number 2 to be used.

NO = 3 both displays to be used.

ICMD - The command word generated by subroutine ASMBLE.

Discussion of flowchart:

Upon entry, NO is checked to insure that it is 1, 2, or 3. If not, an error message is typed out. If NO is 1 or 3, display one is energized and the command word ICMD is transmitted to the display. If NO is 2 or 3, display two is energized and the command word ICMD is transmitted to the display.



FLOWCHART 24

SUBROUTINE DISPLAY

CHAPTER IV

PROPOSALS

Implementing the X-Y Plotter as an Output Device within the Software Package

The X-Y plotter is an analog device. It operates on a DC voltage level. The conversion of analog voltage from digital coding is accomplished by the analog computer interface. Digital to analog converters (DAC's) are used for the conversion.

The X-Y plotter cannot respond to data which changes at computer cycle speeds. Therefore, some method of slowing down to the rate of data transfer must be used.

One method uses two DAC's for the X coordinates and two DAC's for the Y coordinates.

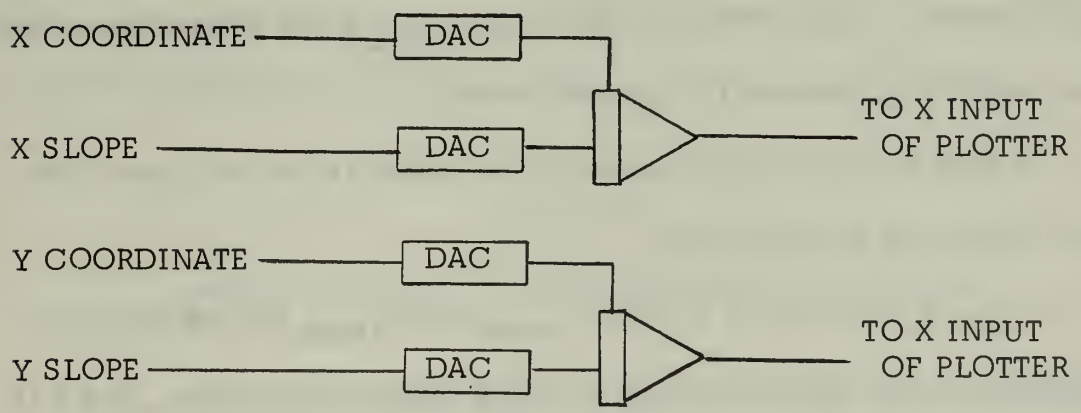


FIGURE 14

FIRST METHOD OF IMPLEMENTATION OF X-Y PLOTTER

In Figure 14 above, the operational amplifiers integrate along the slope between the present coordinate and the next coordinate. The initial condition on the integrator is the present coordinate.

A timing system is made whereby the integrators are in the reset mode, while the present X and Y coordinates are placed on the integrators as initial conditions. The rate of change of the X and Y coordinates is calculated by the digital computer as follows:

$$X_{slope} = \frac{X_{(i+1)} - X_i}{T^*}$$

$$Y_{slope} = \frac{Y_{(i+1)} - Y_i}{T^*}$$

*Where T is the time computer is in compute mode.

As soon as the integrators are placed in the compute mode, the X and Y coordinates are updated to the next values. Thus, when the computer returns to the reset mode the initial conditions are fed to the X - Y plotter. If any errors had developed during the integration period, they would be corrected in the reset mode.

As long as the distance between the points is not too large, the error would not be noticeable.

If a plot of points is desired X_{slope} and Y_{slope} are set equal to zero so that the pen will not move during the compute cycle. The X-Y plotter is connected so that the pen is down in drawing position only during the compute cycle. At the end of the compute cycle, the pen is lifted and the plotter is positioned at the next X and Y coordinate pair.

This method had the advantage of being simple to implement and can make a plot of vectors or points. The entire refresh loop can be copied without operator intervention. Plots of points and vectors can be mixed.

The distinct disadvantage of this method is that the entire analog computer is cycled through the reset-compute cycle.

Another method of implementation makes use of a sampling device. X and Y coordinates are fed to DAC's as fast as the DAC's can convert the information (approximately 50,000 conversions per second). Figure 15 below shows the method of implementation:

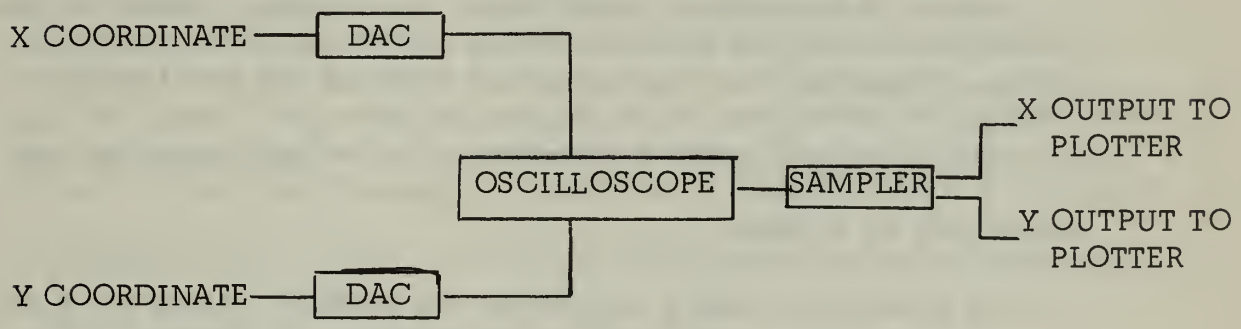


FIGURE 15

SECOND METHOD OF IMPLEMENTING X-Y PLOTTER

The output of the DAC's are fed to an oscilloscope where the curve is displayed. The sampling device takes samples of the voltage on the deflection plates of the oscilloscope. These values are then plotted on the X-Y plotter. When the sampler has completed copying the curve, it generates a signal which can be used to inform the digital computer is has finished.

This method has the advantage that it does not use the analog computer and leaves the analog computer free to perform other tasks. The disadvantages of this method are:

1. The curve on the X-Y plotter will always be a vector plot. There is no provision for point plots.

2. This method is inherently less accurate.

3. Only one curve can be drawn at one time. The entire process must be repeated to draw a second curve.

4. The curves generated have more jitter in them due to the rapid transients generated as the sampler drives the plotter from one sample value to another.

Neither of the methods listed above can reproduce character data strings. Therefore, the subroutine that performs the data transfers must check the mode word in each data string so that character data strings may be skipped.

The input to the DAC's is a fifteen bit quantity. Since and X and Y coordinates in the data strings are only eleven bits, the least significant four bits of the DAC input will be set to zero. The subroutine that implements the X-Y plotter must separate the X and Y components from the data word prior to transmitting them.

Expansion, Contraction, and Translation of Objects Being Displayed

During the display of data it may be desirable to move the presentation horizontally or vertically. This would be accomplished by first designating a point on the display with a light pen and then designating the point to

which we wish to move the display. Once the two coordinates have been designated, simple subtraction will give the change in X and Y coordinates necessary to perform the translation. A subroutine could then add the change in X and Y coordinates to all coordinate points in the data string. The mode word in each data string must be checked to see if it is a character string since only the first word is changed in the character mode.

Any coordinate that is moved outside the eleven bit resolution by the shift will reappear on the opposite side of the screen. (For example, 2048 is the maximum X or Y coordinate that can be plotted and a numerical value of 2050 will appear as a coordinate of two. Thus, anything that moves off the right side of the display will reappear on the left side.) This circular feature can be eliminated by checking each X and Y value after the change coordinate is added to see if it is greater than 2048 or less than 0. If it does exceed either of these limits, it would be set to the appropriate limiting value.

For the expansion and contraction of the display, the light pen is used to designate the point that is to remain fixed while the rest of the display is expanded or contracted. When the coordinates of this point are found, they would be subtracted from each coordinate in the data strings. This sets the coordinates of the point designated to (0,0) so that multiplication by a constant will not change them.

Each coordinate in the data string would then be multiplied by the expansion coefficient or divided by the contraction coefficient. The

original X and Y coordinates of the designated point would then be added to each coordinate pair. All coordinates would then be checked to insure that they fit within the limits of 0 to 2048. If not, they would be forced to 0 or 2048 as above.

When a display picture is expanded, the linearity of values that are forced to the maximum or minimum quantities is destroyed. If the picture is later reduced in size, distortion may result. For this reason, any reduction in size after an expansion should use the user's original data arrays.

If any expansion or contraction is limited to integer quantities, all the mathematics involved would be relatively rapid since the SDS 930 computer has hardwired integer arithimitic.

Core Housekeeping

In computers using a display unit, an area in the memory (buffer) is set aside to contain the information to be displayed. Under actual operating conditions, the buffer contains a certain amount of data. As the display continues, curves (data + strings) are added and deleted. Unless there is some method of core housekeeping, the length of the buffer would be exceeded with the new data while the portions of the buffer that contained the deleted data would be empty.

Continuous Allocation. One possible solution to the core housekeeping problem is the continuous allocation. With this method, the buffer is always filled from the beginning of the buffer area and extends the length of the data to be displayed. As data is deleted, all data is

moved up to fill the recently vacated area. If data is added, the words further away from the starting point are moved down to allow insertion of the new data. This method is efficient in core usage since the buffer does not have any empty areas within the portion being used. It is tremendously inefficient from the time point of view. It takes a great deal of time to move large blocks of words around in memory. This method is not suitable due to this time consideration.

Link Table. The second solution to the core housekeeping is the link method. In this method the housekeeping routine keeps track of empty cells in the buffer. When a string is to be added to the buffer, the words of the data string are placed into any empty buffer areas of four or more words.

Reviewing the data string format:

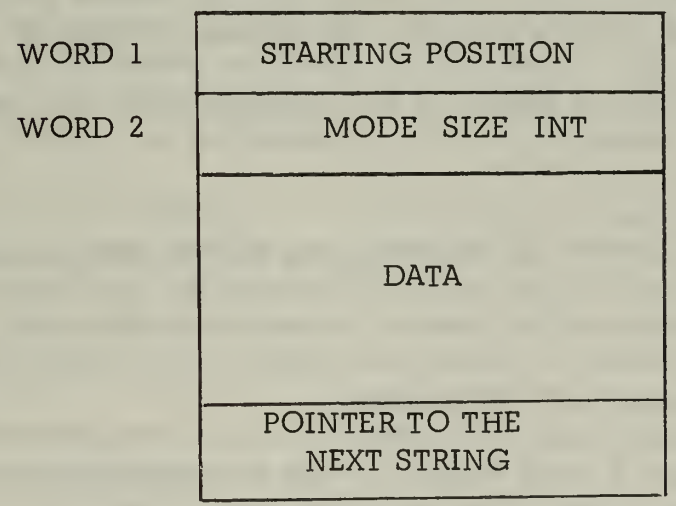


FIGURE 16

DATA STRING FORMAT

It can be seen from the above diagram that the minimum length of a data string is four words; even at this length, there is only one data point. This indicates a worst case efficiency of twenty-five per cent.

If a data string is broken down into several of these small strings, the starting position (WORD 1) must be changed in each sub-string after the first. The starting position of each following sub-string is the last data point in the previous sub-string. The mode word must be copied as the second word in each sub-string. The starting address and length of each sub-string must be generated to be inserted at the end of the previous sub-string as a pointer.

If the data string is in the character mode, the starting position of each sub-string becomes more complicated since it is a function of both the character size and the number of characters which have preceeded it.

In computers having larger word size, the linkage problem is partially alleviated by using a portion of the computer word as a pointer to the next data string.

The software to implement the housekeeping method outlined above becomes quite cumbersome and the memory necessary to store the program offsets any gain in buffer efficiency.

Fixed Record Length. A third method of core housekeeping uses fixed record length. This method requires that all data strings be of some fixed length. Data strings that are shorter than the fixed length

are filled with blanks. Data strings that are longer than the fixed length are broken up into as many records of the set length as needed.

Fixed record length reduces the problem of adding and deleting data strings since each record occupies the same number of computer words. This is not an efficient method of core utilization since there is a certain amount of padding to fill up the records.

This method has been used in many applications where available memory space is not as limited as in the Naval Postgraduate School Computer Laboratory.

Dimension Ordered Chaining. The author prefers a modification of the second method listed. This method used dimension ordered chaining. In this method, the housekeeping routine keeps track of all empty areas of the buffer. When a data string is to be added, the housekeeping routine places the new data string in the smallest area in the buffer in which the string can be fit in one place. After adding the new string to the buffer, the housekeeping routine changes its list of empty areas to reflect the addition. If a data string is deleted, the housekeeping routine does not physically erase the old data string, but changes its empty area list to reflect the deletion. There are two basic assumptions that were used in this housekeeping subroutines:

1. That at some point the housekeeping routine knew exactly what the buffer usage was.
2. When a data string is presented for insertion, the housekeeping routine knows the length of the string.

The first assumption is made true by starting out with an empty buffer and insuring that the housekeeping routine is informed of all changes. The second assumption is valid since the code word associated with each data string contains the length of the string in bits 0 through 8. This method does not have the buffer usage efficiency of the other two methods since small empty areas will be left empty.

It is felt, however, that if an empty area is large enough to affect buffer efficiency, it would be large enough to hold a data string at some time during a display.

The major advantage comes from the fact that the software to implement the subroutine would be relatively simple and fast from the time point of view.

CHAPTER V

APPLICATION EXAMPLE

Chapter I cited an example of how the software package might be used. This chapter will show how the subroutines previously developed would be used.

As stated before, the war game is between a destroyer and a PT boat. The purpose of the war game is to evaluate the effectiveness of various gunnery doctrines against the PT boat.

The conning officer of each ship will be seated at a display unit. He will have analog controls in front of him which change the course and speed of his ship. Display function switches will be designated as "commence firing" and "cease firing" switches. The conning officer of the destroyer will have the following display in front of him: A relative plot of the battle area with the true north pointing up. The destroyer will be at the center of the display. The present and five past positions of the PT boat are plotted representing the information available to him from the combat information center. The present range and true bearing of the PT boat from the destroyer will be displayed. There will be a heading flasher originating at the center of the display area which points in the direction of the ship's heading. When the destroyer fires at the PT boat, shell splashes will be displayed for five seconds.

The PT boat, with its less sophisticated electronics equipment, will have a much more simple display. The plot will have the PT boat at the

center. The present position of the destroyer will be plotted, along with range and bearing of the destroyer. Shell splashes will also be displayed for five seconds.

Timing Considerations

Target range and bearing will be updated at ten second intervals. The time of flight of a round fired by the destroyer will be calculated by the digital computer. This delay equal to the time of flight of the projectile is used prior to displaying the splash.

Information Available

The following information is available:

1. A value corresponding in some manner to the longitude of the destroyer D_x.
2. A value corresponding to the latitude of the destroyer D_y.
3. A value corresponding to the longitude of the PT boat P_x.
4. A value corresponding to the latitude of the PT boat P_y.
5. A range and bearing relative to the destroyer corresponding to the splash point.
6. A signal indicating time a splash occurs.
7. A signal five seconds later indicating the end of the splash.
8. The velocity of the destroyer in X and Y directions DV_x and DV_y.
9. The velocity of the PT boat in X and Y directions PV_x and PV_y.

Subroutine Required

1. Curve add - curve add will be used to add the splash data string to the refresh loop.

2. Unit delete - unit delete will be used to delete information after it is updated.

3. Word - word will be used to display range and bearing.

4. Pack - pack will be used to generate data strings.

5. Past information - past information will be used to display the previous positions of the PT boat.

6. Problem scale - problem scale will be used to obtain floating point scale factors to convert ranges in yards to raster units.

7. Position plot - position plot will be used to generate splash data strings.

8. Assemble - assemble will be used to generate the refresh loop.

9. Information plot - information plot will be used to plot own ship's course and speed.

Method of Implementation

Prior to the start of the war game, the displays are initiated. The range of the destroyer's weapons is less than 30,000 yards, so 60,000 yards will be full scale. Subroutine Problem Scale is called to generate the scale factors. For each display, the origin is fixed at coordinates (1024, 1024). D_x, D_y, P_x, and P_y are read and the necessary mathematics are done so that the range and bearing of the PT boat from the destroyer is known. The symbol Δ is to be used to plot the locations of the PT boat. Position Plot is entered with the information calculated and the data string containing the location of the PT boat is generated. Subroutine Word is called and the range and bearing to the

PT boat is displayed. Subroutine Assemble is called to form the data strings into the refresh loop.

The same procedure is followed to generate the data for the PT boat display.

The problem is now started and the conning officers of each vessel are free to maneuver their ship.

A ten second clock pulse is connected to an interrupt line. This interrupt branches to a "destroyer routine." Another ten second clock pulse (offset from the one above by five seconds) branches to a "PT routine." When the destroyer routine is entered, the present values of D_x, D_y, P_x, P_y, DV_x, and DV_y are read and the following subroutines are called in the order shown:

1. Unit delete removes the data string containing the past position of the PT boat.
2. Position plot generates the new data string containing the new position of the PT boat.
3. Curve add adds the new data string to the refresh loop.
4. Unit delete removes the old range and bearing data string from the refresh loop.
5. Word generates the new range and bearing data string.
6. Curve add adds the new range and bearing data strings to the refresh loop.
7. Past information generates a data string of past positions of the PT boat.

8. Curve add adds the data string generated by Past Information to the refresh loop. (This is necessary only on the first pass through this routine.)

9. Unit delete removes the old course and speed data strings.

10. Information plot generates the new course and speed data strings.

11. Curve add adds the course and speed data string to the refresh loop.

12. Unit delete removes the old heading flasher from the refresh loop.

13. Pack generates the ship's heading vector data string.

14. Curve add adds the ship's heading data string to the refresh loop.

When the "PT routine" is entered the present values of D_x, D_y, P_x, P_y, PV_x, and PV_y are read and the following routines are called in order shown:

1. Unit delete removes the data string containing the past position of the destroyer.

2. Position plot generates the data string containing the new position of the destroyer.

3. Curve add adds the new data string to the refresh loop.

4. Unit delete removes the old range and bearing data string from the refresh loop.

5. Word generates the new range and bearing data string.

6. Curve add adds the range and bearing data string to the refresh loop.

7. Unit delete removes the old heading flasher from the refresh loop.

8. Pack generates the ship's heading vector data string.

9. Curve add adds the ship's heading vector data string to the refresh loop.

When the conning officer of the destroyer depresses the "commence firing" switch, subroutine "shoot" is entered.

Subroutine "shoot" computes the time of flight of the projectile, and the coordinate of the splash point. When an interval equal to the time of flight has passed, subroutine Position Plot generates the splash data string. Curve Add then adds the splash data string to the refresh loop. Five seconds later, the subroutine Unit Delete removes the splash data string from the refresh loop.

Since the firing of a torpedo by the PT boat has no effect on the problem, only the words "torpedo fired" will be displayed when the conning officer of the PT boat depresses the appropriate function switch.

In addition to displaying the data string as they are generated, they are also copied onto magnetic tape. Thus, the problem may be replayed at a later time for analysis with the speed of replay determined by the timing pulses used. The analyst can then speed up the parts of the problem he is not interested in and slow down or even stop the progression of the display for closer inspection of areas of interest.

CHAPTER VI

CONCLUSIONS

The entire software package proposed above will require an estimated 3,000 words of memory. This figure is derived from an estimate of each subroutine averaging 100 words. Some of the subroutines, such as GRID and GRIDIV, will require considerably more than 100 words while most of the interrupt controlled subroutines will require much less than 100 words.

Table 3 below shows the core usage during run time. An average problem has a fixed cell requirement of approximately 12,100 words. This figure leaves 3,900 words for the user's program and display software.

RESIDENT MONITOR	6,600 words
FORTRAN I/O PROCESSOR	2,000 words
PRIMARY LIBRARY	1,000 - 3,000 words (1,500 average)
HYBRID INTERFACE	1,000 - 4,000 words (2,000 average)
DISPLAY SOFTWARE	500 - 3,000 words (1,500 average)

TABLE 3

CORE USAGE DURING RUN TIME

The average display program will probably require approximately 1,500 words of memory for subroutines. This leaves the user with approximately 2,400 words for programs. This number is a bare minimum for programming, but is large enough for the average display program.

The computer configuration is such that the display unit uses the same bank of memory as foreground processing. Cycle stealing will be used by the display unit to retrieve data from memory. Since the memory access rate is determined by the display mode of operation, certain assumptions must be made before discussion of effective speed of computation. Assume that in the character mode, five microseconds are required to draw a character. There are four characters per memory word requiring a memory access every twenty microseconds. This stretches the effective memory cycle from 1.75 microseconds to 2.03 microseconds.

Assuming that in the vector or point mode, no points are more than .1 inch apart. It will take the display unit .5 microseconds to draw the point. This is less than the memory cycle time, so the display unit will trade every other cycle with the central processing unit. This stretches the effective memory cycle to 3.50 microseconds. Table 4 shows the effective cycle speeds and the per cent reduction in speed.

MODE	EFFECTIVE CYCLE SPEED	PER CENT REDUCTION IN EFFICIENCY
CHARACTER	2.03	14%
POINT/VECTOR	3.50	50%

TABLE 4

EFFECTIVE MEMORY CYCLE AND EFFICIENCY WITH DISPLAY UNIT OPERATING

The software package proposed performs most of the basic functions required of a sophisticated software package. Higher level subroutines could be developed which would relieve the user of the necessity of calling each subroutine separately.

The original premise of a versatile system has not been compromised. The example shown in the previous chapter is representative of the complex nature of the display problems; yet all subroutines required for the display were available in the proposed software package.

It is felt that the software package included in this work will relieve the user of all programming necessary to convert his data to the form required for the display unit.

BIBLIOGRAPHY

Allen, Thomas R. and James E. Foote, "Input/output Software Capability for a Man-machine Communication and Image Processing System," Proceedings of the Fall Joint Computer Conference(1964), pp. 387-396.

Ball, N. A. et. al., "A Shared Memory Computer Display System," Institute of Electrical and Electronics Engineers Transactions on Electronic Computers(October, 1966), pp. 750-756.

Jacks, Edwin L., "A Laboratory for the Study of Graphical Man-machine Communication," Proceedings of the Fall Joint Computer Conference (1964), pp. 343-350.

Ninke, William H., "Graphic 1 - A Remote Graphical Display System," Proceedings of the Fall Joint Computer Conference(1965), pp. 839-846.

Scientific Data Systems, Letter to US Naval Postgraduate School (October 5, 1966).

Scientific Data Systems, Memo to US Naval Postgraduate School(July, 1966).

Scientific Data Systems, SDS Fortran IV Reference Manual, Santa Monica, California(October, 1966).

Scientific Data Systems, SDS 930 Computer Reference Manual, Santa Monica, California(February, 1967).

Scientific Data Systems, SDS 920/930 Computer Programmed Operator's Technical Manual, Santa Monica, California(July, 1965).

Scientific Data Systems, SDS Real-time Monitor Reference Manual, Santa Monica, California(July, 1967).

Scientific Data Systems, SDS Symbol and Meta-symbol Reference Manual, Santa Monica, California(August, 1967).

Scientific Data Systems, "Specifications, Display Console for US Naval Post Graduate School,"(May, 1967).

Sutherland, Ivan E., "Sketchpad - A Man-machine Graphical Communication System," Proceedings of the Spring Joint Computer Conference(1963), pp. 329-346.

Terlet, R. H., "The CRT Display Subsystem of the IBM 1500 Instruction System," Proceedings of the Fall Joint Computer Conference(1967), pp. 169-176.

GLOSSARY

Buffer	- A portion of digital computer memory reserved for a certain purpose such as display data.
Command word	- A twenty-four bit computer word which represents a data string in memory.
Cycle stealing	- Two independent units of the computer, each having memory addressing capability take turns accessing a portion of memory.
Data string	- A group of memory cells containing either coordinate pairs or alphanumeric data to be displayed.
Mode word	- The second word in a data string. It contains information specifying the type of data, the intensity of display, and the size of characters in the data string.
Raster unit	- The smallest increment of distance on the display area. It is usually specified as a fraction of the total display width or height.
Refresh loop	- One or more data strings connected by command words which will be displayed at one time.
Starting address	- The absolute location in computer memory where the first word in a data string is stored.
Word count	- The number of computer words making up a data string.

APPENDIX A

SDS 930 COMPUTER MACHINE INSTRUCTION LIST

CODE	MNEMONIC	NAME
00	HLT	HALT
01	BRU	BRANCH UNCONDITIONALLY
02	EOM	ENERGIZE OUTPUT M
06	EOD	ENERGIZE OUTPUT TO DIRECT ACCESS CHANNEL
12	MIW	MEMORY INTO W BUFFER WHEN EMPTY
13	POT	PARALLEL OUTPUT
14	ETR	EXTRACT
16	MRG	MERGE
17	EOR	EXCLUSIVE OR
20	NOP	NO OPERATION
23	EXU	EXECUTE
32	WIM	W BUFFER INTO MEMORY WHEN FULL
33	PIN	PARALLEL INPUT
35	STA	STORE A
36	STB	STORE B
37	STX	STORE X
40	SKS	SKIP IF SIGNAL NOT SET
41	BRX	INCREMENT INDEX AND BRANCH
43	BRM	MARK PLACE AND BRANCH

CODE	MNEMONIC	NAME
46	---	REGISTER CHANGE
50	SKE	SKIP IF A EQUALS M
51	BRR	RETURN BRANCH
52	SKB	SKIP IF M AND B DO NOT COMPARE ONES
53	SKN	SKIP IF M NEGATIVE
54	SUB	SUBTRACT
55	ADD	ADD M TO A
56	SUC	SUBTRACT WITH CARRY
57	ADC	ADD WITH CARRY
60	SKR	REDUCE M, SKIP IF NEG
61	MIN	MEMORY INCREMENT
62	XMA	EXCHANGE M AND A
63	ADM	ADD A TO M
64	MUL	MULTIPLY
65	DIV	DIVIDE
66	---	SHIFT RIGHT
67	---	SHIFT LEFT
70	SKM	SKIP IF A = M ON B MASK
71	LDX	LOAD INDEX
72	SKA	SKIP IF M AND A DO NOT COMPARE ONES
73	SKG	SKIP IF A GREATER THAN M
74	SKD	DIFFERENCE EXPONENTS AND SKIP

CODE	MNEMONIC	NAME
75	LDB	LOAD B
76	LDA	LOAD A
77	EAX	COPY EFFECTIVE ADDRESS INTO INDEX REGISTER

APPENDIX B

SYSTEM INTERRUPT ASSIGNMENTS AND EOM CODES

INTERRUPT FUNCTION	ASSIGNED MEMORY LOCATION
Display console #1 channel end	0203
Display console #1 function/keyboard	0204
Display console #1 light pen	0205
Display console #2 channel end	0206
Display console #2 function/keyboard	0207
Display console #2 light pen	0210

EOM FUNCTION	EOM COMMAND
Address display #1	EOM 31001
Address display #2	EOM 31002
Enable display #1 interrupts	EOM 31110
Enable display #2 interrupts	EOM 31120
Read display #1 address	EOM 31101
Read display #2 address	EOM 31102
Read display #1 function/keyboard	EOM 31111
Read display #2 function/keyboard	EOM 31112
*Test for display #1 ready	EOM 31400
*Test for display #2 ready	EOM 32400

*will skip next instruction if addressed display is ready

APPENDIX C

LIST OF ALLOWABLE CHARACTERS FOR DISPLAY UNIT

DISPLAY UNIT CHARACTER	ASC II CODE	CORRESPONDING SDS CHARACTERS	SDS INTERRUPT CODE
△	00	△	57
A	01	A	21
B	02	B	22
C	03	C	23
D	04	D	24
E	05	E	25
F	06	F	26
G	07	G	27
H	10	H	30
I	11	I	31
J	12	J	41
K	13	K	42
L	14	L	43
M	15	M	44
N	16	N	45
O	17	O	46
P	20	P	47
Q	21	Q	50
R	22	R	51
S	23	S	62

DISPLAY UNIT CHARACTER	ASC II CODE	CORRESPONDING SDS CHARACTERS	SDS INTERRUPT CODE
T	24	T	63
U	25	U	64
V	26	V	65
W	27	W	66
X	30	X	67
Y	31	Y	70
Z	32	Z	71
[	33	[	35
\	34	\	76
]	35	]	55
↑	36	↑	37
EOL	37	NONE	
SPACE	40	SPACE	12 or 60
!	41	!	52
"	42	∩	75
#	43	+	77
$	44	$	53
%	45	TAB	72
√	46	√	17
'	47	'	14
(	50	(	74
)	51	)	34
*	52	*	54

DISPLAY UNIT CHARACTER	ASC II CODE	CORRESPONDING SDS CHARACTERS	SDS INTERRUPT CODE
+	53	+	20
,	54	,	73
-	55	-	40
.	56	.	53
/	57	/	61
0	60	0	0
1	61	1	1
2	62	2	2
3	63	3	3
4	64	4	4
5	65	5	5
6	66	6	6
7	67	7	7
8	70	8	10
9	71	9	11
:	72	:	15
;	73	;	56
<	74	<	36
=	75	=	13
>	76	>	16
?	77	?	32

APPENDIX D

METHOD OF GENERATING SPECIAL CHARACTERS

The viewing area is 2048 units in both the horizontal and vertical directions. The size of the characters must be defined in terms of raster units. For example: A resistor is to be drawn. It is to be one inch long by one-fourth inches wide.

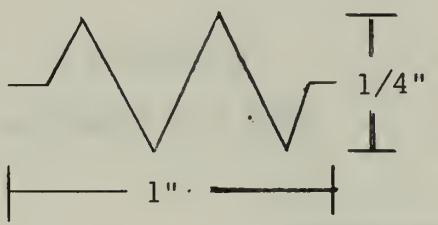


FIGURE 17

RESISTOR

Horizontal direction:

2048 raster units = 18 inches

1 inch = 113 raster units(100 rounded off)

Vertical direction:

2048 raster units = 13 inches

1 inch = 157 raster units

1/4 inch = 39 raster units(40 rounded off)

Draw grid, superimpose resistor:

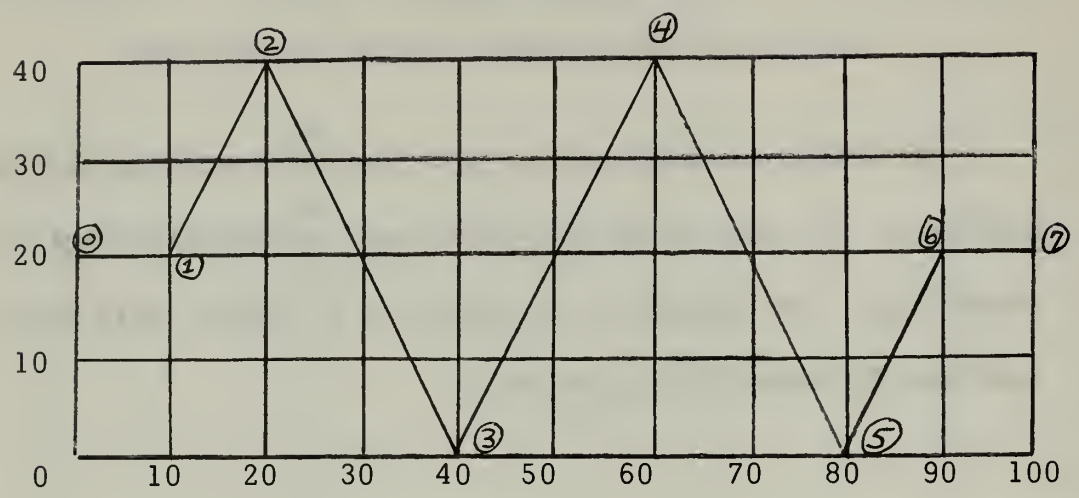


FIGURE 18

GRID AND RESISTOR

List the coordinates:

POINT	X_{10}	Y_{10}	X_8	X_8
0	0	20	0	24
1	10	20	12	24
2	20	40	24	50
3	40	0	50	0
4	60	40	74	50
5	80	0	120	0
6	90	20	132	24
7	100	20	144	24

TABLE 5

COORDINATES

Assemble into data string. (On next page)

This is now a firm data string. Before it is displayed, the intensity must be added to octal positions 6 and 7 of WORD 2. The X and Y coordinates must be added to each word except WORD 2.

Assemble into data string:

WORD NO.	OCTAL REPRESENTATION								REMARKS
	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	
WORD 1	0	0	0	0	0	0	5	0	POSITION WORD SAME AS WORD 3
WORD 2	0	0	0	0	1	1	0	0	MODE, SIZE, AND INTENSITY WORD
WORD 3	0	0	0	0	0	0	5	0	STARTING COORDINATE POINT 0
WORD 4	0	0	2	4	0	0	5	0	POINT 1
WORD 5	0	0	5	0	0	1	2	0	POINT 2
WORD 6	0	1	2	0	0	0	0	0	POINT 3
WORD 7	0	1	7	0	0	1	2	0	POINT 4
WORD 8	0	2	4	0	0	0	0	0	POINT 5
WORD 9	0	2	6	2	0	0	5	0	POINT 6
WORD 10	0	3	1	0	0	0	5	6	POINT 7

TABLE 6

DATA STRING ASSEMBLY

Example #2: Draw a capacitor.

It is to be one-half inch high and one inch long.

FIGURE 19

CAPACITOR

Horizontal direction:

2048 raster units = 18 inches

1 inch = 113 raster units (100 rounded off)

Vertical direction:

2048 raster units = 13 inchez

1/2 inch = 79 raster units (80 rounded off)

Draw grid, superimpose capacitor:

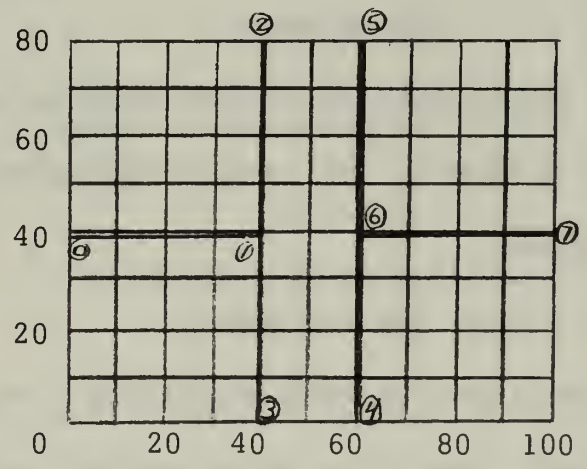


FIGURE 20

GRID AND CAPACITOR

Note: In this case, the figure is not entirely connected, (Point 3 to Point 4) by setting the Bit 23 of the computer word equal to one, the display moves on to the next point, but does not draw a vector.

List the coordinates:

POINT	X_{10}	Y_{10}	BLANK	X_8	Y_8
0	0	40	0	0	50
1	40	40	0	50	50
2	40	80	0	50	120
3	40	0	0	50	0
4	60	0	1*	74	0
5	60	80	0	74	120
6	60	40	0	74	50
7	100	40	0	144	4

TABLE 7

COORDINATES

*Note: Point 3 is not connected to Point 4; therefore, no vector is to be drawn from Point 3 to Point 4. This is accomplished by inserting a "1" in the "BLANK" column.

Assemble into data string. (On next page)

The octal numbers do not seem to correspond to the 8 X and Y coordinates. Actually, this is due to the fact that the X coordinates is packed into binary bits 0 through 10, and the Y coordinates are packed into bits 12 through 22.

Assemble into data string:

WORD NO.	OCTAL REPRESENTATION								REMARKS
	0_0	0_1	0_2	0_3	0_4	0_5	0_6	0_7	
WORD 1	0	0	0	0	0	0	2	0	POSITION WORD SAME AS WORD 3
WORD 2	0	0	0	0	1	1	0	0	MODE, SIZE, & INTENSITY WORD
WORD 3	0	0	0	0	0	1	2	0	STARTING COORDINATE POINT 0
WORD 4	0	1	2	0	0	1	2	0	POINT 1
WORD 5	0	1	2	0	0	2	4	0	POINT 2
WORD 6	0	1	2	0	0	0	0	0	POINT 3
WORD 7	0	1	7	0	0	0	0	1	POINT 4*
WORD 8	0	1	7	0	0	2	4	0	POINT 5
WORD 9	0	1	7	0	0	1	2	0	POINT 6
WORD 10	0	3	1	0	0	1	2	0	POINT 7

Note: 0_7 = 1 means no vector from WORD 6 to WORD 7.

TABLE 8

DATA STRING ASSEMBLY

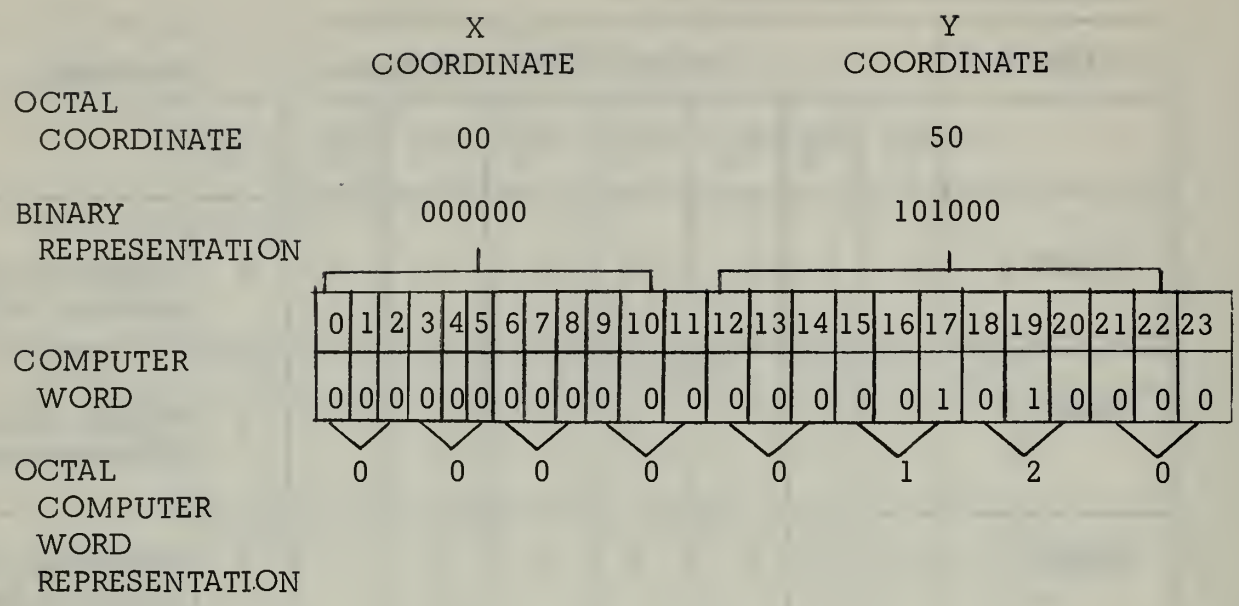


FIGURE 21

PLACEMENT OF X AND Y COORDINATES IN COMPUTER WORD

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library U. S. Naval Postgraduate School Monterey, California 93940	2
3.	Commandant U. S. Coast Guard Headquarters Washington, D. C.	1
4.	Prof. Mitchell Cotton Department of Electrical Engineering U. S. Naval Postgraduate School Monterey, California 93940	3
5.	Lt. William Douglas Bechtel, USCG Coast Guard Radio Station Alexandria, Virginia	1
6.	Capt. Robert G. Von Elm c/o Commander, Fifth Coast Guard District Portsmouth, Virginia 23704	2
7.	Mr. Robert L. Limes Department of Electrical Engineering Computer Facility U. S. Naval Postgraduate School Monterey, California 93940	1

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b GROUP

3. REPORT TITLE

O LINE GRAPHICAL DISPLAY SYSTEM

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Thesis

5 AUTHOR(S) *(Last name, first name, initial)*
BECHTEL, William D., Lieutenant, U. S. Coast Guard

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1968	130	14

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10 AVAILABILITY/LIMITATION NOTICES

[redacted]

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

On-line graphical display of data of various types offers a powerful tool for the visual analysis of information. A software system is proposed to implement on-line display in a general purpose hybrid simulation laboratory. The subroutines required to implement this are put forward. Specific proposals are made in areas related to the basic software package. An application example is included to show the interrelationship of the software package.

DD FORM 1 JAN 64 1473

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
ON-LINE COMPUTATION						
GRAPHICAL DISPLAY						
ON-LINE DISPLAY						
DIGITAL COMPUTERS						